Single- and Multiple-Objective Optimization with Differential Evolution and Neural Networks

Man Mohan Rai*
NASA Ames Research Center, Moffett Field, CA-94035, USA

INTRODUCTION

Genetic and evolutionary algorithms[1] have been applied to solve numerous problems in engineering design where they have been used primarily as optimization procedures. These methods have an advantage over conventional gradient-based search procedures because they are capable of finding global optima of multi-modal functions (not guaranteed) and searching design spaces with disjoint feasible regions. They are also robust in the presence of noisy data. Another desirable feature of these methods is that they can efficiently use distributed and parallel computing resources since multiple function evaluations (flow simulations in aerodynamics design) can be performed simultaneously and independently on multiple processors. For these reasons genetic and evolutionary algorithms are being used more frequently in design optimization. Examples include airfoil and wing design[2 - 3] and compressor and turbine airfoil design. They are also finding increasing use in multiple-objective and multidisciplinary optimization.[4] The references cited here represent a very small sample of the literature.

One problem with genetic and evolutionary algorithms is that they often require many more function evaluations than other optimization schemes to obtain the optimum. In fact they are not the preferred method when a purely local search of a smooth landscape is required. Rai[5] presents an evolutionary method, based on the method of Differential Evolution[6] (DE), and investigates its strengths in the context of some test problems as well as nozzle and turbine airfoil design. The results of applying a neural network-based response surface method (RSM [7]) to the same design problems are also presented in this study. It was found that DE required about an order of magnitude more computing time than the neural network-based design method. In a more recent article Madavan[8] has explored the possibility of combining DE with local search methods and, utilized the resulting hybrid method in airfoil inverse design. The best variant of these combined methods required 420 function evaluations for this inverse design. In contrast, this inverse design problem required about 50 simulations with a neural-network based algorithm[9]. Here again, the computational cost is about an order of magnitude less. In general where applicable, significant cost reductions can be achieved by using gradient- and RSM-based methods instead of evolutionary algorithms. However, the latter approach is preferred for multi-modal functions and design spaces with disjoint feasible regions. One of the pioneers of evolutionary algorithms (EAs), Scwefel[10], writes with regard to choosing between optimization methods (in particular EAs and local search methods) "…there cannot exist but one method that solves all problems effectively as well as efficiently. These goals are contradictory."

Multiple-objective design optimization is an area where the cost effectiveness and utility of evolutionary algorithms (relative to local search methods) needs to be explored. Deb[11] presents numerous evolutionary algorithms and some of the basic concepts and theory of multi-objective optimization. Mietinnen[12] also presents an excellent survey of the state of the art in multiple-objective optimization. Both these authors provide a large number of references for the interested reader.

The objective here is to introduce a relatively new evolutionary method, Differential Evolution (DE), developed by Price and Storn [6]. In its original version [6], DE was developed for single objective optimization. DE is a population-based method for finding global optima. It is easy to program and use and requires relatively few user-specified constants. These constants are easily determined for a wide class of problems. Fine-tuning the constants will yield the solution to the optimization problem at hand more rapidly. The method can be efficiently implemented on parallel computers and can be used for

*Senior Scientist, Exploration Technology Directorate

continuous, discrete and mixed discrete/continuous optimization problems. It does not require the objective function to be continuous and is noise tolerant. Additionally, the method does not require the transformation of continuous variables into binary integers. The basic method is presented later in the text. Although DE is an effective and efficient global optimization method compared to other evolutionary and genetic algorithms, in general, powerful local search methods continue to be the best choice for locating local optima. Here we also explore the possibility of integrating DE with response surface methodology; the objective being a hybrid design procedure that has the strengths of both methods [5].

Differential evolution can also be used effectively in multiple-objective optimization. Abbas et al.[13] first proposed an extension to DE (PDE) to handle multiple objectives. PDE is a Pareto-based approach that uses non-dominated ranking and selection procedures to compute several Pareto-optimal solutions simultaneously. Madavan[14] presents a different extension to DE to handle multiple objectives. This method is also a Pareto-based approach that uses non-dominated ranking and selection procedures to compute several Pareto-optimal solutions simultaneously. It combines the features of DE and the NSGA-II method of Deb et al.[15].

In more recent articles, Rai[16-17] presents an evolutionary algorithm, based on the method of DE, for multiple-objective design optimization. One goal of this developmental effort was a method that required a very small population of parameter vectors to solve complex multiple-objective problems involving several Pareto fronts (global and local) and nonlinear constraints. Applications of this evolutionary method to some difficult model problems involving the complexities mentioned above are also presented in these articles. The computed Pareto-optimal solutions closely approximate the global Pareto-front and exhibit good solution diversity. Many of these solutions were obtained with small population sizes. Here we present Rai's extension of DE to multiple-objective optimization and apply it to numerous model problems.

Achieving solution diversity and accurate convergence to the exact Pareto front usually requires a significant computational effort with evolutionary algorithms. Here we explore the possibility of using neural networks to obtain estimates of the Pareto optimal front using non-dominated solutions generated by DE as training data. Neural network estimators have the potential advantage of reducing the number of function evaluations required to obtain solution accuracy and diversity, thus reducing cost to design. The estimating curve or surface can be used to generate any desired distribution of Pareto optimal solutions.

SINGLE-OBJECTIVE DIFFERENTIAL EVOLUTION

The single-objective evolutionary algorithm proposed by Rai[5] draws upon ideas from several genetic algorithms and evolutionary methods. One of them is a relatively new member to the general class of evolutionary methods called differential evolution [6]. As with other evolutionary methods and genetic algorithms, DE is a population based method for finding global optima. The three main ingredients are mutation, recombination and selection. Much of the power of this method is derived from a very effective mutation operator that is simple and elegant. Mutations are obtained by computing the difference between two randomly chosen parameter vectors in the population and adding a portion of this difference to a third randomly chosen parameter vector to obtain a candidate vector. The resulting magnitude of the mutation in each of the parameters is different and close to optimal. For example, in the case of an elliptical objective function in two dimensions, the set of all possible mutation vectors would be longer in the direction of the major axis and shorter in the direction of the minor axis. Thus, the mutation operator adapts to the particular objective function and this results in rapid convergence to the optimal value. In addition, this approach automatically reduces the magnitude of mutation as the optimization process converges.

To describe one version of single-objective DE [6], we consider the set of parameter vectors at the nth generation, $\mathbf{X}_{j,n}$. The subscript j refers to the jth member in a population of N parameter vectors and,

$$\mathbf{X}_{j,n} = \left[\mathbf{x}_{1,j,n}, \mathbf{x}_{2,j,n}, \ldots . \mathbf{x}_{D,j,n}\right] \qquad (1)$$

where $x_{i,j,n}$ corresponds to the parameter value in the ith dimension in a D-dimensional problem. The initial population is assumed to be randomly distributed within the lower and upper bounds specified for each dimension. The mutation, recombination and selection operators are then applied to the population of parameter vectors as many times as required. To evolve the parameter vector $X_{j,n}$ we randomly pick three other parameter vectors $X_{a,n}$, $X_{b,n}$ and $X_{c,n}$ such that $a \neq b \neq c \neq j$. A trial vector Y is then defined as

$$Y = X_{a,n} + F(X_{b,n} - X_{c,n}) \tag{2}$$

where F is a user specified constant, $(0 < F < 1)$. The candidate vector $Z = [z_1, z_2, ...z_D]$ is obtained via a recombination operator involving the vectors $X_{j,n}$ and Y and is defined as

$$Z = \begin{cases} y_i & \text{if } r_i \leq C \\ x_{i,j,n} & \text{if } r_i > C \end{cases} \tag{3}$$

where r_i is a uniformly distributed random variable $(0 \leq r_i < 1)$ and C is a user specified constant, $(0 < C < 1)$.The final step in the evolution of $X_{j,n}$ involves the selection process and, for the minimization of the objective function $f(X)$, is given by

$$X_{j,n+1} = \begin{cases} Z & \text{if } f(Z) \leq f(X_{j,n}) \\ X_{j,n} & \text{if } f(Z) > f(X_{j,n}) \end{cases} \tag{4}$$

In other words, the selection process involves a simple replacement of the original parameter vector with the candidate vector if the objective function decreases by such an action.

Aerodynamic shape optimization varies from simple unimodal function optimization to multimodal function optimization, constrained optimization, optimization in cases where the search space contains disjoint regions of feasibility, and multiple-objective optimization. The performance of the evolutionary method used in this study was first investigated using test cases with some of these attributes. These cases are discussed below.

Unconstrained Optimization of a Multimodal Function

One of the most desirable attributes of evolutionary search algorithms is their ability to locate the global optimum of a multimodal function. Although this is not guaranteed, in many practical situations they tend to produce better solutions than purely local searches of the parameter space. The first test problem was chosen to highlight this particular aspect of DE. The function to be minimized is two-dimensional and is given by

$$f(x, y) = \left\{ 0.002 + \sum_{n=1}^{25} \left[n + (x - a_n)^6 + (y - b_n)^6 \right]^{-1} \right\}^{-1} \tag{5}$$

The constants a_n in Eq. 5 are given by

$$a_1 = -32,\ a_2 = -16,\ a_3 = 0,\ a_4 = 16,\ a_5 = 32$$
$$a_n = a_{n \bmod 5} \qquad n = 6, 7, ...25 \tag{6}$$

and the constants b_n are given by

$$b_{1,2,..5} = -32$$
$$b_{6,7,..10} = -16$$
$$b_{11,12,..15} = 0 \quad (7)$$
$$b_{16,17,..20} = 16$$
$$b_{21,22,..25} = 32$$

The lower and upper bounds for the search are as follows:

$$-65.536 \le x, y \le 65.536 \quad (8)$$

The function given in Eq. 5 is also referred to as De Jong's fifth function or Shekel's foxholes[6]. It has 25 minima in the region $-32.0 \le x, y \le 32.0$. The function is nearly constant everywhere except near the minima. Figure 1 shows contours of this function in the region $-40.0 \le x, y \le 40.0$. The minimum nearest the lower left hand corner of the square is the global minimum $(f \approx 0.998004)$.

Twenty parameter vectors were used in the search process. Figure 2 shows the reduction in the objective function with increasing number of function evaluations (representative of the cost of optimization). This data was obtained as an average over 10 optimization runs with different initial parameter vector populations. All the test runs converged to the global optimum. Convergence to the optimum was defined by the criterion $f(x, y) \le 0.998004$. The evolutionary method used here required 680 steps to converge to the optimum. Price and Storn[6] required 672 steps for convergence.

Constrained Optimization

Many engineering optimization problems are constrained by equality and inequality constraints that can be linear or nonlinear. The second test case was chosen to evaluate the ability of the current evolutionary method in solving such constrained optimization problems. This optimization problem represents the design of a gearbox (the Golinski speed reducer[18]). It consists of one objective and 11 inequality constraints, of which 7 are nonlinear. There are seven variables and seven associated upper and lower bounds. The objective is to minimize the function

$$f(x_1, x_2, ..x_7) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \quad (9)$$

subject to the inequality constraints

$$27.0 - x_1x_2^2x_3 \le 0 \quad (10)$$

$$397.5 - x_1x_2^2x_3^2 \le 0 \quad (11)$$

$$1.93x_4^3 - x_2x_3x_6^4 \le 0 \quad (12)$$

$$1.93x_5^3 - x_2x_3x_7^4 \le 0 \quad (13)$$

$$\left[\left(745.0x_4 / (x_2x_3)\right)^2 + 16.9 \times 10^6\right]^{1/2} - 110.0x_6^3 \le 0 \quad (14)$$

$$\left[\left(745.0x_5 / (x_2x_3)\right)^2 + 157.5 \times 10^6\right]^{1/2} - 85.0x_7^3 \le 0 \quad (15)$$

$$x_2x_3 - 40.0 \le 0 \quad (16)$$

$$5.0x_2 - x_1 \leq 0 \quad (17)$$

$$x_1 - 12.0x_2 \leq 0 \quad (18)$$

$$1.5x_6 - x_4 + 1.9 \leq 0 \quad (19)$$

$$1.1x_7 - x_5 + 1.9 \leq 0 \quad (20)$$

The lower and upper bounds for the search are as follows:

$$2.6 \leq x_1 \leq 3.6 \quad (21)$$

$$0.7 \leq x_2 \leq 0.8 \quad (22)$$

$$17 \leq x_3 \leq 28 \quad (23)$$

$$7.3 \leq x_4, x_5 \leq 8.3 \quad (24)$$

$$2.9 \leq x_6 \leq 3.9 \quad (25)$$

$$5.0 \leq x_7 \leq 5.5 \quad (26)$$

Note that x_3 is an integer variable but is treated as a continuous variable. This approach works because the optimal value for this variable corresponds to its lower bound and, this lower bound is an integer.

Ten parameter vectors were used in the search process. The method yielded the following values for the seven variables:

$x_1 = 3.5000008$

$x_2 = 0.7000000$

$x_3 = 17$

$x_4 = 7.3000002$

$x_5 = 7.7153251$

$x_6 = 3.3502148$

$x_7 = 5.2866545$

The corresponding function value is **f = 2994.36** and, this compares well with the result of Azarm and Li[18] of 2994.0. All the constraints are satisfied at the optimal location. Four of them are active constraints. Figure 3 shows the mean convergence rate, which was generated by averaging the rates obtained in ten different runs of the method; each with a different initial population of parameter vectors. The minimum is obtained to within 0.1% in about 600 function evaluations.

Search Spaces with Multiple Feasible Regions

In aerodynamic design and other optimization problems, the search space may contain multiple regions of feasibility embedded within infeasible regions. A global search method must be able to find

the global optimum and perhaps several of the local optima in the feasible regions and prioritize them. The third test case was chosen to investigate the method's ability (DE[5,6]) to solve such problems.

The problem is defined as maximizing the two-dimensional function

$$f(x, y) = x^2 + y^2 \tag{27}$$

The search region is defined by the constraints

$$-5.0 \le x, y \le 5.0 \tag{28}$$

and the feasible regions are further constrained by

$$0.1 - \sum_{n=1}^{n=nmax} \exp\left\{-K_n\left[(x - x_n)^2 + (y - y_n)^2\right]\right\} \le 0 \tag{29}$$

These constraints (Eqs. 28 and 29) together yield multiple regions of feasibility that are disjoint but are contained within a square. The number of feasible regions is determined by the parameter nmax, but is not necessarily equal to nmax because some of them coalesce to form larger regions of feasibility. Figure 4 shows the regions of feasibility obtained for nmax = 42 and random choices for x_n and y_n. Each of the exponential terms (in isolation), in the summation given in Eq. 29, yields a circular region. Together they yield nearly circular regions of feasibility that are either separate or merge with others. The "radii" of these nearly circular regions are determined by the constants K_n. The large region at the center of the square was generated with K = 1.5 and all the other regions were generated with K = 20.

The value of the objective function increases monotonically from the center of the square outward. The contours of this function are circles centered at the origin (center of the square). Every feasible region in Fig. 4 has at least two contour circles that are tangent to it; one at the smallest possible radius and one at the largest possible radius. The function minimum and maximum, in the given region of feasibility, lie at these smallest and largest radius values, respectively, at which tangency occurs.

Ten parameter vectors were used in the search process. They yielded three different maximum values including the global maximum. The locations of these maxima are indicated by the square symbols in Fig. 4. The number 1 designates the global maximum, and the numbers 2 and 3 designate the other two local maxima in descending order. Interestingly, just ten parameter vectors yielded three different maxima. This facet of the method may be critical in making a design trade-off. For example, the global maximum may not be achievable with current manufacturing processes, and the designer may have to choose one of the local maxima.

Design optimization with DE

The application of DE to aerodynamic design is relatively straightforward. The aerodynamic shape of interest is first parameterized using an appropriate method. The prudent selection of geometry parameters is one of the most critical aspects of any shape optimization procedure. Variations of the aerodynamic shape can be obtained by varying these parameters. Geometrical constraints imposed for various reasons, such as structural, aerodynamic (e.g., to eliminate flow separation) should be included in this parametric representation as much as possible. Additionally, the smallest number of parameters should be used to represent the aerodynamic shape. The second step involves the specification of the upper and lower bounds for the geometry parameters to be used in the search process. This step typically involves some knowledge of the aerodynamics involved and constraints such as the maximum and minimum thickness of an airfoil (from structural considerations or to prevent choking of the flow in a turbine). The third step involves defining an appropriate objective function (a function of the geometric

parameters) to be minimized. A given engineering objective can be achieved using different objective functions; some more difficult than others to optimize. In some cases the search for an optimum can be made significantly easier by using the appropriate objective function. Hence it behooves the designer to spend some time on making the appropriate choice of objective function. The final step involves using DE to determine the optimal set of geometric parameters. Applications of evolutionary algorithms to aerodynamic design will be covered by other lecturers in this course and will not be emphasized here.

Hybridization of DE and local search methods

Genetic and evolutionary algorithms often require many more function evaluations than other optimization schemes to obtain the optimum. In fact they are not the preferred method when a purely local search of a smooth landscape is required. There have been numerous attempts to hybridize population based search methods and local search methods to create new methods with the best properties of the constituents. There are two commonly used hybridization techniques. The first one consists of creating response surfaces that are then searched by genetic and evolutionary algorithms. The second incorporates a local search technique within the GA/EA that replaces members of the population with better individuals obtained via a limited local search. This replacement can happen at random or at a specified frequency. Here we present a simple hybridization technique that is particularly suited for optimization problems involving disjoint feasible regions and/or multimodal functions. The purpose is to illustrate the utility of hybrid techniques by combining DE and a response surface method (RSM) based on neural networks[5].

We illustrate the method using an aerodynamic design optimization study that involves the shape optimization of a supersonic nozzle. The nozzle pressure for the area distribution

$$\mathbf{A(x) = 0.35(2.0 + x^2),} \qquad \mathbf{0 \le x \le 1.0} \tag{30}$$

was obtained from a simple one-dimensional analysis (inlet Mach number of 2.0). The computed pressure values at 21 equally spaced points in the region $\mathbf{0 \le x \le 1.0}$ were provided to the design procedure as target values. The objective was to recover the nozzle cross-sectional area at these axial locations, $\mathbf{A_i}$ **(i = 1, 2,.....21).** This constitutes a 21 dimensional optimization problem for the evolutionary method. Parameterization of the shape of the nozzle will certainly reduce the dimensionality of the search space, but this approach was not pursued.

The optimization problem as stated above can be solved rapidly in just a few generations with DE. Here the problem is modified to make it considerably more difficult to solve in order to test the method's ability to locate the global optimum in the presence of disjoint regions of feasibility. Instead of searching for optimal values of A_i within a hypercube, we define

$$\mathbf{A_i = r_i \sin(4\pi\, r_i)} \tag{31}$$

and search for the optimal values of r_i in the hypercube defined by

$$\mathbf{0.25 \le r_i \le 1.25, \quad 1 \le i \le 21} \tag{32}$$

The optimal values of A_i lie between 0.70 and 1.05 and, consequently the optimal values of r_i lie between 1.0 and 1.25. The regions defined by the inequalities, $\mathbf{0.25 \le r_i \le 0.50}$ and $\mathbf{0.75 \le r_i \le 1.00}$ are infeasible because they yield non-positive values of area. Equation 31 yields positive values of area in the region $\mathbf{0.50 \le r_i \le 0.75}$, however this region does not contain the global optimum. Figure 5 shows the feasible regions in the $\mathbf{r_1 - r_2}$ plane for a case with only two variables A_1 and A_2. There are four feasible regions and the one closest to the top right hand corner contains the global optimum. The 21 dimensional hypercube considered here has $\mathbf{2^{21}}$ feasible regions with the global optimum contained in one of them. The feasible region containing the global optimum occupies only a miniscule fraction of the total search volume $\mathbf{(1/4^{21})}$.

DE was applied to this optimization problem. The sum-of-squares objective function was defined as defined as:

$$\mathbf{SSE} = \sum_{i=1}^{21} (P_i - p_i)^2 \qquad (33)$$

where P_i is the target pressure and p_i is the pressure at the same axial location for any given nozzle shape. Sixty parameter vectors were used in the search process. Figure 6 shows the optimal (as obtained by the evolutionary method) and the exact area distributions in the axial direction. The two are in close agreement with each other demonstrating the success in finding the global optimum in this case with 2^{21} feasible regions. Figure 7 shows the corresponding optimal and exact pressure distributions in the nozzle.

As indicated earlier there are considerable advantages to developing a hybrid aerodynamic design process that possesses the best attributes of both DE and the neural-network based method. In the current nozzle-design case, it would be difficult for the user to provide the neural network based method with an initial geometry that lies in the feasible region containing the global optimum. Here we use DE to obtain such an initial geometry for the neural network based method and then subsequently use the rapid convergence properties of the latter method to search for the minimum. In general the transfer of control from one method to another will be based on heuristics (such as an order of magnitude reduction in the objective function value) and may need to be repeated a couple of times at different stages of the evolutionary process. The overall computational cost may still be a fraction of the cost associated with a purely evolutionary approach.

In this study, for the purpose of illustration, the parameter vector corresponding to the lowest value of the objective function at each generation is identified. This “best” parameter vector is used to generate the initial nozzle design when it first arrives in the feasible region containing the global optimum. Obviously this approach is not feasible in the general case. It is used here only to depict the best-case scenario where the transfer of control is optimal. A heuristic method such as picking the best parameter vector after a certain number of generations, or after every order-of-magnitude reduction in the objective function, would transfer control later in the evolutionary process than the current optimal transfer of control. However, such a “heuristics-based’” transfer of control would not require information regarding the position of the best parameter vector relative to the feasible region containing the global optimum.

Figures 6 and 7 show the initial geometry and corresponding pressure distribution supplied to the neural network based system. These were obtained from DE when the best parameter vector first entered the feasible region containing the global optimum. This initial geometry was then transformed into the optimal geometry using the neural network based method. Figure 8 is a plot of the convergence rate and shows the variation of the objective function with the number of nozzle flow solutions used in the optimization process. The purely evolutionary method required 60000 function evaluations to reduce the objective function by about 7 orders of magnitude. The hybrid method with optimal transfer of control requires about 20% as many function evaluations. The rapid convergence obtained with the neural network based scheme is particularly noteworthy. Only 185 nozzle flow solutions were required by this scheme to reduce the value of the objective function from approximately 1.0 to 4×10^{-7}. This indicates that the evolutionary process can be tapped several times for a solution that lies within the feasible region containing the global optimum without incurring a large penalty.

Figure 9 shows the results of tapping DE for the best parameter vector after 100 and 500 generations (heuristic approach). The neural network based RSM converges to a local optimum when it is initialized with the best parameter vector obtained at the end of 100 generations. This is because the initial design supplied to it is in a feasible region that does not contain the global optimum and, consequently the sum-of-squares error quickly asymptotes to a rather large value. The evolutionary process yields an initial design that lies in the feasible region containing the global optimum when it is tapped after 500 generations. In this case the neural network based method rapidly yields the global optimum (92 nozzle flow solutions for a 10 ten-order of magnitude reduction in SSE). Thus the hybrid method halves the number of flow solutions required for design optimization. The hybrid approach as described above is also applicable to cases where the function is multimodal.

MULTIPLE-OBJECTIVE DIFFERENTIAL EVOLUTION

Abbas et al.[13] first proposed an extension to DE (PDE) to handle multiple objectives. PDE is a Pareto-based approach that uses non-dominated ranking and selection procedures to compute several Pareto-optimal solutions simultaneously. The population of parameter vectors is first sorted to obtain the non-dominated set. Mutation and recombination is undertaken only among members of the non-dominated set. The resulting candidate vector replaces a member of the population if it dominates the first selected parent ($\mathbf{Z}$ replaces $\mathbf{X}_{j,n}$ if it dominates $\mathbf{X}_{a,n}$). When the total number of parameter vectors in the non-dominated set exceeds a threshold value, a distance metric in parameter space is used to remove members of this set that are in close proximity. This feature improves solution diversity.

In a more recent study Madavan[14] presents a different extension to DE to handle multiple objectives. This method is also a Pareto-based approach that uses non-dominated ranking and selection procedures to compute several Pareto-optimal solutions simultaneously. It combines the features of DE and the NSGA-II method of Deb et al.[15]. The main difference between DE (single-objective) and this method lies in the selection process. New candidate vectors obtained from mutation and recombination are simply added to the population thus resulting in a population that is twice as large. This larger population is subjected to the non-dominated sorting and ranking procedure of Deb et al. The ranking is then used to subsequently reduce the population to its original size. Solution diversity is achieved by ascribing diversity ranks to members of the last non-dominated set that contributes to the new population. Diversity ranks are based on the crowding distance metric proposed by Deb et al. Unlike the distance metric in PDE, this crowding distance metric is computed in objective space.

An important issue that both these studies (Abbas[13] et al. and Madavan[14]) do not address is the manner in which single-objective DE extracts valuable mutation information directly from the population of vectors, and the retention of this ability in the context of multi-objective optimization. As explained by Price and Storn[19], under the assumption that the parameter vectors of a population are distributed around a level line in parameter space that represents their mean value, the set of vectors created by vector differences (in the mutation operator) are close to optimal. For example, when the contours of the objective function are elliptic, the set of all possible vector differences is longer in the direction of the major axis and shorter in the direction of the minor axis. In the presence of a second populated minimum the set of vector differences include ones that facilitate the transfer of parameter vectors from the proximity of one minimum to the proximity of the other, thus making DE a powerful global optimizer.

However, multiple-objective optimization requires the entire Pareto optimal front to be adequately populated. Consider a situation where the Pareto-optimal front is highly curved in parameter space and the parameter vectors are distributed evenly along this front but do not coincide with it. Clearly vector differences involving vectors from disparate regions of this front are not very effective mutation vectors. However, parameter vectors that straddle the Pareto front in parameter space and are in close proximity to each other will yield mutations (vector differences) that are more likely to result in superior candidate vectors. The parent vectors as well as the vector being considered, $\mathbf{X}_{a,n}$, $\mathbf{X}_{b,n}$, $\mathbf{X}_{c,n}$ and $\mathbf{X}_{j,n}$ need to be in proximity for effective mutation and recombination. This is especially true in the final stages of optimization. In the initial stages of optimization the entire front can be considered a single entity in a basin of attraction, being approached from afar by the parameter vectors. Localization of the relevant vectors used in mutation and recombination may not be necessary at this early stage.

The methods of Abbas et al.[13] and Madavan[14] have yielded accurate Pareto-optimal fronts in some model problems without localization. This is most likely due to the presence of a population of vectors and corresponding vector differences, some of which are appropriate mutations. Additionally, in cases where the Pareto-optimal front is not very curved in parameter space, localization may not be an issue. However, both the studies report that better convergence was achieved with a value of F (Eq. 2) around 0.3 for the model problems considered. This is about 1/3 to 1/2 of the value normally used in single-objective DE-based optimization and is indicative of the need for localization. Typical values of F used with current method lie between 0.6 and 1.0 for the first 75% of the total number of generations, followed by $F \leq 0.6$ for the remaining generations. The reduction in F towards the end of the evolutionary process results in a small improvement in convergence and quality of Pareto-optimal solutions. This improvement is to be expected because unlike single objective DE where the magnitude of the mutation vectors approaches zero as the parameter vectors converge to a point, the mutation vectors in the case of multiple objective

optimization remain finite even after all the parameter vectors are located on the Pareto optimal front. The population of vectors continues to redistribute itself on the Pareto optimal front to obtain better solution diversity. In fact a significant number of generations may be required to obtain superior solution diversity. Pareto-optimal solutions for some of the cases presented later in the text were obtained rapidly with large values of F (5.0 to 10.0). The common feature in these problems was that the Pareto front was a subset of the boundary of the search region. The large values of F used in the first part of the evolution accelerate the movement of the population from the interior to the relevant part of the boundary in these cases.

Rai's[17] extension of DE to multiple-objective optimization consists of the following steps:

(1) Determine the set of non-dominated parameter vectors (rank one only as in PDE[13], and unlike NSGA-II).

(2) Reduce this set of potential parent vectors to improve solution diversity if the number of parameter vectors in this set exceeds a certain threshold value. The method used to perform this operation is discussed below (as in PDE, and unlike NSGA-II).

(3) For each member of the population, chose three parent vectors from the non-dominated set, compute a candidate vector as in Eqs. 2 and 3 and, add this candidate vector to the bottom of the list of parameter vectors to create a population that is twice as large as the original (as in NSGA-II, and unlike PDE).

(4) Identify the non-dominated set of vectors and perform a bubble-sort so that the new set of non-dominated vectors move to the top of the list. This automatically pushes those that are no longer non-dominated down the list (different from NSGA-II and PDE).

(5) Retain only the first N parameter vectors (as in NSGA-II and unlike PDE).

This method (like PDE) only requires the rank-one non-dominated vectors to be determined. Hence it easier to program than NSGA-II and the computational expense for identifying the pool of parent vectors is also less than that required by NSGA-II. The selection process is similar to that of NSGA-II and is hence more elitist than PDE. Tests on some complex multi-objective optimization problems have demonstrated that the procedure described above is a powerful multi-objective optimization tool.[16-17]

Localization in this method is achieved in the following manner; given the parameter vector $\mathbf{X_{j,n}}$ from the population size of N, the parent vector $\mathbf{X_{a,n}}$ is chosen as

$$\mathbf{X_{a,n} = X_{i,n} \text{ if } r \le 1 - d_{i,j}/d_{max}} \qquad \mathbf{(34)}$$

where $\mathbf{X_{i,n}}$ is randomly chosen from the non-dominated set, $\mathbf{r}$ is a uniformly distributed random variable $\left(\mathbf{0 \le r < 1}\right)$, $\mathbf{d_{i,j}}$ is the distance between the vectors $\mathbf{X_{i,n}}$ and $\mathbf{X_{j,n}}$ in parameter space, and $\mathbf{d_{max}}$ is the maximum distance between parameter vectors in the population. Equation 34 states that the vector $\mathbf{X_{i,n}}$ is more likely to be chosen for small values of $\mathbf{d_{i,j}}$ (relative to $\mathbf{d_{max}}$). The parent vectors $\mathbf{X_{b,n}}$ and $\mathbf{X_{c,n}}$ are obtained similarly. Clearly Eq. 34 does not preclude the possibility of distant vectors being chosen as parents, it merely gives preference to parent vectors that are in proximity to $\mathbf{X_{j,n}}$.

One problem that occurs when inadequate population sizes are used is stagnation, a situation where the population stops evolving[19]. The method presented by Rai[17] resorts to a second mutation operator to maintain a healthy evolutionary process when very small population sizes are used. In its simplest version this involves adding a random variation to one of the coordinates of a candidate vector. The candidate vector, the coordinate to be perturbed, the magnitude of the mutation operator and the generation of occurrence are all chosen randomly. Parameter mutations are specified as

$$\mathbf{p_m = KR(2r - 1)/2} \qquad \mathbf{(35)}$$

where R is the linear magnitude of the search space in the coordinate that is being perturbed, K is a user specified constant (typically between 0.1 to 0.25), and r is a uniformly distributed random variable

$(0 \le r < 1)$. The presence of the term **(2r - 1)** makes the method relatively insensitive to the choice of larger values of K. This mutation operator was found to be an effective tool in preventing stagnation and premature convergence in the context of small populations. Mutations involving the simultaneous perturbation of several coordinates of the candidate vector can be devised using this principle.

Solution diversity is achieved using a method that bears some resemblance to that of PDE. The mutation vectors are computed using all of the non-dominated members of the population. When the number of non-dominated vectors exceeds a certain threshold value (in this study the threshold value was set to the original population size), members of this set are eliminated in a sequential manner. In the first pass the two vectors that are closest to each other (this can be determined in either parameter space or objective space) are identified. The vector among these two that is *further* down the list of vectors is tagged for removal. The process is continued until the number of non-dominated members is equal to the specified threshold value. The more select members of the original non-dominated population are then used for the crossover operation and to define the new population. Experimentation with a method similar to that of NSGA-II of obtaining solution diversity did not perform as well in some problems. The evolutionary pressure exerted by this approach is subtle and, perhaps not as effective in cases with Pareto-fronts having regions that are difficult to populate.

The performance of the multiple-objective version of DE presented here is now investigated using test cases that include Pareto fronts with different attributes. One of the cases exhibits multiple local Pareto fronts. Both unconstrained and constrained problems are solved.

Unconstrained Multiple-Objective Optimization

Several unconstrained multi-objective optimization cases are solved here. Deb[11] presents a detailed discussion of these cases. They are constructed to test the ability of the optimization method to converge to the global Pareto front and, compute Pareto fronts that are convex, non-convex and discontinuous. The first four cases were formulated by Zitzler, Deb and Thiele[20], and are denoted as ZDT1, ZDT2, ZDT3 and ZDT4. The fifth test case was first proposed by Viennet[21], and is labeled as VNT1. All the ZDT test cases involve two objective functions, whereas VNT1 involves three objective functions. The test cases of Zitzler et al.[20] that are solved here can be formulated as

$$\begin{aligned} &\textbf{Minimize}: f_1(X) \\ &\textbf{Minimize}: f_2(X) = g(X)\, h(f_1(X), g(X)) \end{aligned} \qquad (36)$$

where X is a vector in n-dimensional parameter space and the functions $f_1(X)$, g(X) and h(X) are defined differently for each case. The type of problem complexity (non-convex Pareto front, discontinuous Pareto front, multiple Pareto fronts etc.) as well as the degree of complexity can be specified by appropriate choices of the functions $f_1(X)$, g(X) and h(X). The global Pareto front for all these cases is g(X) =1.0.

The test case ZDT1 has thirty variables and is defined by the following functions

$$\begin{aligned} f_1(X) &= x_1 \\ g(X) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \\ n &= 30 \\ 0 &\le x_i \le 1 \end{aligned} \qquad (37)$$

The Pareto front for this case is convex. The multi-objective evolutionary algorithm presented here and ten parameter vectors were used to obtain the Pareto front. Figure 10 shows the computed Pareto optimal solutions and the exact Pareto front. The agreement between the two is good. *It should be noted*

that DE usually requires between two and 100 times as many parameter vectors as the number of variables in the problem. The optimal ratio depends on the complexity of the optimization problem. Here the solution is obtained with 10 parameter vectors for a 30 variable problem in 250 generations. The computed solutions also exhibit good solution diversity, that is, the computed Pareto optimal points are nearly evenly spaced and cover the entire front.

The test case ZDT2 also has thirty variables and is defined by the following functions

$$\begin{aligned} f_1(X) &= x_1 \\ g(X) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(f_1, g) &= 1 - (f_1/g)^2 \\ n &= 30 \\ 0 &\le x_i \le 1 \end{aligned} \qquad (38)$$

It is more complex than ZDT1 because the Pareto front is non-convex. The problem was solved using 10 parameter vectors and 250 generations. Figure 11 shows the computed Pareto optimal solutions and the exact Pareto front. The computed solutions are again in close agreement with the exact Pareto front and exhibit good solution diversity. The test case ZDT3 is a 30 variable problem and is defined as follows

$$\begin{aligned} f_1(X) &= x_1 \\ g(X) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ h(f_1, g) &= 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1) \\ n &= 30 \\ 0 &\le x_i \le 1 \end{aligned} \qquad (39)$$

An important characteristic of ZDT3 is that the Pareto front is discontinuous in objective space. Forty parameter vectors and 300 generations were used to solve this problem. Although 10 parameter vectors yielded accurate Pareto optimal solutions, their density along the Pareto front was inadequate. As seen in Fig. 12, the computed solutions are in close agreement with their exact counterparts and exhibit good solution diversity. ZDT3 requires a moderately large population size because of the length and complexity of the Pareto optimal front and the simultaneous requirement that this front be adequately populated.

The test case ZDT4 has 10 variables and is a particularly difficult problem for all multi-objective optimization methods because it exhibits numerous local Pareto fronts[11]. The global and the next best local Pareto front are given by g(X) = 1.00 and g(X) = 1.25, respectively. The problem is defined as

$$\begin{aligned} f_1(X) &= x_1 \\ g(X) &= 1 + 10(n-1) + \sum_{i=2}^{n}\left(x_i^2 - 10\cos(4\pi x_i)\right) \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \\ n &= 10 \\ 0 &\le x_1 \le 1 \\ -5 &\le x_{2,3,\ldots,10} \le 5 \end{aligned} \qquad (40)$$

Inadequate population sizes generally yield Pareto optimal solutions on one of the local Pareto fronts of ZDT4. The global Pareto optimal solutions have been found to be elusive to capture in previous studies by other investigators. Figure 13 shows the computed Pareto optimal solutions obtained with the current

method and, the exact global Pareto front. Six parameter vectors and 3333 generations were used to compute the Pareto optimal solutions. Both, proximity to the exact front and solution diversity are good. This computation required 0.68 CPU seconds on a single 400MHz SGI (MIPS) processor. The rather unusual number of generations (3333) was picked to determine if the present method could yield the Pareto optimal solutions with about 20000 function evaluations. Clearly this objective has been met. Figure 14 shows the Pareto optimal solutions for ZDT4 obtained in five consecutive runs. All of them converge to the global optimum thus demonstrating the reliability of the method.

The test case VNT1 involves three objective functions and two variables. VNT1 is defined as

$$\begin{aligned}
&\text{Minimize: } f_1(x_1, x_2) = 0.5\left(x_1^2 + x_2^2\right) + \sin\left(x_1^2 + x_2^2\right) \\
&\text{Minimize: } f_2(x_1, x_2) = 15.0 + \left(3x_1 - 2x_2 + 4\right)^2 / 8.0 + \left(x_1 - x_2 + 1\right)^2 / 27.0 \\
&\text{Minimize: } f_3(x_1, x_2) = -1.1\exp\left(-\left(x_1^2 + x_2^2\right)\right) + 1.0 / \left(x_1^2 + x_2^2 + 1\right) \\
&-3.0 \le x_1, x_2 \le 3.0
\end{aligned} \tag{41}$$

The Pareto front is discontinuous in both design space as well as objective space. Figure 15 shows the computed Pareto optimal solutions obtained with a population size of 50 in 200 generations. Although accurate solutions were obtained with much smaller population sizes, 50 parameter vectors were used for this problem to better represent the rather complex Pareto front. Figure 15 shows the projection of the Pareto front on the (f_1, f_3) plane. Again, the current evolutionary method yields optimal solutions that are diverse and are close to the exact optimal front in spite of the complexity of this front as well as the three-dimensionality of the objective space.

As mentioned earlier, the parent vectors ($\mathbf{X}_{a,n}$, $\mathbf{X}_{b,n}$, $\mathbf{X}_{c,n}$ and $\mathbf{X}_{j,n}$ in Eqs. 1, 2 and 3) need to be in proximity to recapture the essence of DE in a multi-objective setting. ZDT1, ZDT2, ZDT3, and ZDT4 are all characterized by Pareto-fronts that are straight lines in parameter space. The Pareto-optimal front for VNT1 in parameter space, can be closely approximated by two straight lines. None of these cases are significantly affected by localization. Although their Pareto fronts seem complicated in objective space, they are simple in parameter space.

The following problem (MMR1), exhibits attributes that make it suitable to illustrate the need for localization. MMR1 is defined as

$$\begin{aligned}
&\text{Minimize: } f_1(x_1, x_2) = 0.5x_1^2 + 0.5\sin^2\left(0.5\pi x_2\right) \\
&\text{Minimize: } f_2(x_1, x_2) = 0.5\left(x_1 - 1.0\right)^2 + 0.5\left(x_2 - 1.0\right)^2 \\
&-2.5 \le x_1, x_2 \le 2.5
\end{aligned} \tag{42}$$

Figure 16 shows the computed and exact global Pareto-optimal fronts in parameter space. The computed solutions were obtained with 100 parameter vectors in 250 generations. The Pareto front is highly curved and has a branch point thus making it a good candidate to test the ideas on localization discussed earlier. It should be noted that the Pareto-optimal solutions on the upper and lower branches with the same x_1-coordinate have identical function values and hence occupy the same location on the Pareto-front in objective space. Hence, in order to adequately populate the Pareto-front in objective space, it is sufficient if the combined population (upper and lower branches) is adequate in a given segment $a < x_1 < b$. As seen in Fig. 17 the computed Pareto-optimal solutions in objective space are in close agreement with the exact Pareto-optimal front and, exhibit good solution diversity.

Figure 18 shows the convergence rates obtained with four variants of the method; 1) No localization and a constant value of F=1.0, 2) Localization and a constant value of F=1.0, 3) No localization and a value of F=0.3, and 4) Localization and a reduction in the value of F from 1.0 to 0.3 after 75% of the total number of generations. The results of 20000 independent runs were averaged to obtain the data depicted in this figure. Clearly, the first three cases result in larger errors and show signs of approaching an

asymptotic value. The best results are obtained with localization and a reduction in F in the last few generations of the run. The result thus obtained is about three times more accurate than those obtained with the second and third variants of the method and about seven times more accurate than the solution obtained with the first variant of the method. Additionally, the fourth variant of the method shows a continued decrease in the error after 200 generations.

Constrained Multiple-Objective Optimization

Many engineering problems are constrained by equality and inequality constraints that are linear or nonlinear. A novel technique of constraint satisfaction in the context of single-objective evolutionary methods was implemented for differential evolution by Rai [5].This approach to constraint satisfaction is also applicable to the multi-objective differential evolution algorithm. The following simple example (labeled MMR2) is used to demonstrate the constraint handling ability of the method. The feasible region consists of the interior of three nearly circular sub-regions. MMR2 is defined as:

$$\begin{aligned}
&\text{Minimize } f_1(x_1, x_2) = 0.5\left(x_1^2 + x_2^2\right) \\
&\text{Minimize } f_2(x_1, x_2) = 0.5\left(x_1 - 1\right)^2 + 0.5\left(x_2 - 1\right)^2 \\
&\text{Subject to } g\left(x_1, x_2\right) \le 0.0 \\
&\qquad g\left(x_1, x_2\right) = 0.5 - \sum_{j=1}^{3} \exp\left(-r_j^2\right) \\
&\qquad r_j^2 = 15.0\left(x_1 - x_1^j\right)^2 + 15.0\left(x_2 - x_2^j\right)^2 \qquad (43) \\
&\qquad \left(x_1^1, x_2^1\right) = (0.0, 0.0) \\
&\qquad \left(x_1^2, x_2^2\right) = (0.5, 0.5) \\
&\qquad \left(x_1^3, x_2^3\right) = (1.0, 1.0) \\
&\qquad -2.0 \le x_1, x_2 \le 2.0
\end{aligned}$$

Figure 19 shows the computed Pareto front obtained with 30 parameter vectors in 40 generations (3639 function evaluations), and the exact Pareto front for both the constrained and unconstrained cases. The Pareto front for the constrained case consists of three unconnected segments. The computed optimal solutions are in good agreement with the exact Pareto front and good solution diversity has been achieved. Figure 20 shows the segmented Pareto front and the constraint boundaries in parameter space. The computed optimal solutions satisfy the constraint (lie within the circles), and are close to the exact Pareto front.

Neural Network Estimates for Pareto Optimal Fronts

From the examples presented above it is clear that the current method is capable of yielding accurate Pareto optimal solutions with very small population sizes. Figure 13 shows the computed Pareto optimal solutions obtained for ZDT4 with 6 parameter vectors. The computed optimal solutions are well dispersed and the end points of the Pareto front are captured. However, the distribution of solutions is not uniform. Continued evolution does result in better solution diversity. Larger population sizes also tend to yield better solution diversity. Both of these approaches require additional function evaluations. It has been observed that in many cases the parameter vectors converge fairly quickly to the Pareto optimal front and then the population of vectors continues to redistribute itself on this front to yield better solution diversity. In fact a significant number of generations may be required to obtain superior solution diversity. One approach to eliminate the cost involved in obtaining good solution diversity during the evolutionary process is to use an estimation technique to fit the data obtained from DE (or any other evolutionary method). The response surface thus obtained can be used directly or to generate the required

distribution of Pareto optimal solutions. Here we explore the use of neural networks in obtaining accurate, uniformly distributed Pareto optimal solutions.

Feed-forward artificial neural networks are essentially nonlinear estimators. They can be used to approximate complex multi-dimensional functions without having to specify an underlying model. Training a neural network to model data requires determining the connection weights that define the network. Nonlinear optimization methods are typically employed to obtain these weights. The connection weights are not uniquely defined; many different sets of weights may yield acceptably low training error for a given network architecture and dataset. This multiplicity of acceptable weight vectors can be used to advantage. One could select the neural network (or equivalently the corresponding set of weights) with the smallest validation error if validation data is available. In constructing response surfaces which approximate the Pareto optimal front using Pareto optimal solutions as training data, this approach requires setting aside some of these solutions as validation data. It is reasonable to expect the generalization ability of the set of weights selected in this manner to be superior compared to the rest of the sets of weights. However, this approach results in very few training data when the number of optimal solutions is small as in Fig. 13.

In the absence of validation data, multiple trained neural networks can be effectively utilized by creating a hybrid network.[22 - 23] The output of the hybrid network can be defined as a simple average of the outputs of all the trained neural networks. It can be shown[22] that the sum-of-squares error (SSE) thus obtained (or integrated squared error) in modeling the function underlying the data, is a factor of N less than the average SSE, where N is the number of trained networks. The essential assumption that is made to obtain this result is that the errors produced by the different networks have zero mean and are not correlated. When the errors produced by the different networks are correlated the SSE of the hybrid network continues to be less than or equal to the average SSE. However, it is not necessarily reduced by a factor of N. Note that the networks used in this ensemble average do not have to possess the same architecture or even be trained on the same training set.

A second and more general way of combining the outputs of different trained networks is to weight the output of each network such that error of this combined output is minimized.[23] Given weights α_i (i = 1,...N) , the optimal weights can be obtained by minimizing the function

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j C_{ij} \qquad (44)$$

subject to the constraint

$$\sum_{i=1}^{N}\alpha_i = 1 \qquad (45)$$

The matrix C in Eq. 44 is the error correlation matrix. Details of this method of creating a hybrid network are discussed by Perrone and Cooper.[23] Unlike the simple averaging technique, this method does not explicitly require that the mean error for the networks be zero, or, that the network models be mutually independent. However, practical considerations such as maintaining the full rank of C, and imposing the constraint $\alpha_i \geq 0$ to prevent data extrapolation, once again require these assumptions to be met. Hybrid networks have been used effectively to construct response surfaces for design optimization[9, 24]. The more general ensemble approach yielded a better model only in some of the cases investigated.

The creation of a hybrid network serves to reduce the variance. Effective hybridization requires the neural network training method to yield numerous uncorrelated network models. The nonlinear optimization process used to train each network can be started with different random weights to accomplish this task. Methods that improve the generalization ability of the individual networks, such as regularization and network architecture optimization can be embedded at this level.

The generalization ability of hybrid networks was tested using low-order polynomials and a Gaussian function in Rai[9]. Good generalization (both in the region where data was available and outside of this region) was obtained. A feed-forward network with two hidden layers and the more general ensemble method of creating the hybrid network (Eqs. 44-45) were used in all these cases. These encouraging results led to the investigation of the generalization accuracy that can be obtained for higher-order polynomials, polynomials combined with other functions, polynomials in multiple dimensions, and cases where the training data is contaminated by noise.[24] Excellent generalization was obtained in all these cases. These investigations have resulted in better training algorithms for feed-forward neural networks.

Figure 21 shows the generalization obtained for a fifth-order polynomial modeled using eight training points and a hybrid network consisting of ten single hidden layer neural networks. The full line was obtained with the neural network and the dashed line (superimposed on the full line) was obtained using the exact function. Neural network generalization is excellent throughout the region $-2 \le x \le 3$ although training data is available only in the region $0 \le x \le 1$. The ability of the hybrid network to extrapolate is evident in this case. Clearly this does not constitute proof that hybridization will always work as well (especially in the extrapolation mode). Note that a simple polynomial fit (fifth-order) would yield perfect generalization. However, it is equally important to note that the network was not supplied with the information that the function underlying the training data was a polynomial.

The neural network generalization shown in Fig. 21 is surprisingly accurate even in the extrapolation mode. A natural question to ask at this point is why the network generalization closely approximates the original function used to generate the training data, given that there are many curves that would fit this data. The answer lies in the fact that the given data (8 points) when interpolated using a set of polynomial basis functions (for example the Legendre polynomials) uniquely define the original curve, assuming a convention such as interpolating the data with the lowest-order polynomial. Hence, the neural network will reproduce the original function to the extent that it can mimic the polynomial basis functions. Neural networks can be made to mimic some classes of functions either through the choice of network connectivity or preprocessors that process the input data before they are fed to the input nodes.

The results of Fig. 21 indicate that it may be possible to use hybrid networks to obtain accurate estimates of a Pareto-optimal front given a few Pareto-optimal solutions. To investigate this possibility we consider the following two-objective optimization problem (labeled MMR3):

$$\text{Minimize: } f_1(X) = \frac{\sum_{n=1}^{D} \left((x_n + 1)/n\right)^2}{\sum_{n=1}^{D} (2/n)^2}$$

$$\text{Minimize: } f_2(X) = \frac{\sum_{n=1}^{D} (x_n - 1)^2}{4D} \qquad (46)$$

where D is the dimensionality of the search space. The Pareto optimal front for this problem is given by

$$x_n = \frac{(n^2 + 1)x_1 + (n^2 - 1)}{(n^2 - 1)x_1 + (n^2 + 1)} \qquad n = 2, 3, \ldots D$$

$$-1 \le x_1 \le 1 \qquad (47)$$

MMR3 was formulated so that the individual objectives mimic those found commonly in engineering optimization (contours of the first function form multi-dimensional ellipsoids in parameter space) and, to

generate a Pareto optimal front that is not a straight line (unlike ZDT1 – ZDT4). MMR3 also scales easily to any number of dimensions; the ratio of the major to the minor axes of the ellipsoids being equal to D.

In the first test case Pareto optimal solutions for MMR3 were obtained with DE for D = 2. Eleven parameter vectors were used in this computation. The Pareto optimal solutions thus obtained were sorted (increasing value of f_1) and every alternate one was provided to the hybrid network as training data (six training pairs). The training data are not uniformly distributed. The hybrid network consisted of ten single-hidden-layer feed-forward networks. The input to the networks was the first coordinate in parameter space (x_1) and the output was the second coordinate (x_2). Obtaining the estimating curve/surface in parameter space (as opposed to objective space) permits the generation of additional Pareto optimal solutions in a straightforward manner. Figure 22 shows the Pareto-optimal front in parameter space obtained from the hybrid neural network. The training data and the exact Pareto optimal front are also provided in Fig. 22. The estimated front and the exact Pareto front are nearly identical. The non-uniformity of the Pareto optimal solutions computed using DE is not an issue because the estimating curve can be used to generate any distribution of optimal solutions. This is an example of the savings that can be realized by not pursuing superior diversity in solutions generated by evolutionary methods.

In this computation the Pareto-optimal solutions were fully converged and thus the training data was free of noise. Additionally the end points of the Pareto front were captured in the evolutionary process. This case represents a typical application of hybrid networks to obtain an estimate of the Pareto optimal front. Hybrid networks can also be used in cases where the training data are contaminated with noise and do not include the boundary points of the Pareto front. For moderately noisy data requiring a moderate amount of extrapolation the hybridization and training methods of Rai[9, 24] can be used to generate the estimating curve. Extremely noisy data and situations where large extrapolations are required call for a more effective, specialized hybridization principle.

One such principle has been developed and applied effectively for MMR3. Here we provide some examples of this methodology. Figure 23 shows the training data obtained from DE for MMR3 with D = 2. The computed solutions are not fully converged and exhibit a considerable amount of noise. The exact Pareto optimal front and the estimate obtained from the hybrid network are also shown in this figure and are nearly identical. The ability of the hybrid network in extracting an accurate estimate of the Pareto front from the noisy data is evident in this figure. The new method of hybridization does require additional function evaluations. The relative costs incurred in continued refinement of the solutions using DE and, generating estimating surfaces will be discussed in Rai[25]. Figure 24 shows results for a case where the DE search was confined to a portion of the region containing the Pareto optimal front. The training data thus obtained only cover a portion of the Pareto front. The hybridization technique yields an estimated front that is once again in close agreement with the exact front both in the region of the data (interpolative mode) and far removed from the data (extrapolative mode).

Figure 25 shows the results of a similar exercise with MMR3 for the case D=4. Here three hybrid networks, each consisting of ten individual networks, were used to generate an estimate of the Pareto optimal front. The first hybrid network was used to represent the functional relationship between x_1 and x_2, the second between x_3 and x_2 and, the third between x_4 and x_2. The variable x_2 was used as the primary variable. The arc length along the Pareto front could have been used as the primary variable instead. Figure 25 shows the training data, the exact relationship between these variables and the corresponding neural network estimates. The training data were once again obtained in a restricted search space and hence do not cover the entire front. It is evident from the figure that the estimates are accurate in both interpolative and extrapolative modes. The three estimates can also be obtained using a single hybrid network in which x_2 is the input and x_1, x_3 and x_4 are obtained as the output of the individual networks. Such an approach may be necessary for more complex Pareto front topologies.

In conclusion, basic single-objective DE and the modifications that are required to create an effective multiple-objective optimization algorithm are presented here. While localization and the sequential diversity enhancing operators have been developed primarily for DE, they should find use in

other GA/EA based multiple objective optimization algorithms. Localization has the potential of enhancing the crossover operation that is prevalent in such algorithms. Application of RSM and evolutionary algorithms to multiple-objective optimization is in its infancy and much more research is required to determine the range of applicability of these methods.

REFERENCES

1. Goldberg, D. E., *Genetic algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

2. Obayashi, S., and Tsukahara, T., "Comparison of Optimization Algorithms for Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 35, No. 8, August 1997, pp. 1413-1415.

3. Holst, T. L., and Pulliam, T. H., "Aerodynamic Shape Optimization Using a Real Number Encoded Genetic Algorithm," AIAA Paper no. 2001-2473, AIAA 19th Applied Aerodynamics Conference.

4. Obayashi, S., and Yamaguchi, Y., "Multi-objective Genetic Algorithm for Multi-disciplinary Design of Transonic Wing Platform," *Journal of Aircraft* , Vol. 34, No. 5, 1997, pp. 690-693.

5. Rai, M. M., "Towards a Hybrid Aerodynamic Design Procedure Based on Neural Networks and Evolutionary Methods," AIAA Paper No. 2002-3143, AIAA 20th Applied Aerodynamics Conference, St. Louis Missouri, June 24-26, 2002.

6. Price, K., and Storn, N., "Differential Evolution," *Dr. Dobb's Journal*, April 1997, pp. 18-24.

7. Myers, R. H., and Montgomery, D. C., *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley and Sons, New York, 1995.

8. Madavan, N. K., "Aerodynamic Shape Optimization Using Hybridized Differential Evolution," AIAA Paper No. 2003-3792, 21st Applied Aerodynamics Conference, Orlando, Florida, June 23-26, 2003.

9. Rai, M. M., "A Rapid Aerodynamic Design Procedure Based on Artificial Neural Networks," AIAA Paper No. 2001-0315, AIAA 39th Aerospace Sciences Meeting, Reno, Nevada, Jan. 8-11, 2001.

10. Schwefel, H.-P. "Advantages (and Disadvantages) of Evolutionary Computation Over Other Approaches", *Handbook of Evolutionary Computation*, Institute of Physics Publishing and Oxford University Press, 1997

11. Deb, K., *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, 2001.

12. Miettinen, K. M., *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, 2002.

13. Abbas, H. A., Sarker, R., and Newton, C., "PDE: A Pareto-Frontier Differential Evolution Approach for Multi-objective Optimization Problems," Proceedings of the Congress on Evolutionary Computation, 2001, Vol.2, pp. 971-978, Piscataway, New Jersey, May 2001.

14. Madavan, N. K., "Multiobjective Optimization Using a Pareto Differential Evolution Approach," *Proceedings of the Congress on Evolutionary Computation*, 2002, Vol.2, pp. 1145-1150, Honolulu, Hawaii, May 2002.

15. Deb, K. Agrawal, S., Pratap, A., Meyarivan, T., “A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II”, Proceedings of the Parallel Problem Solving from Nature VI Conference, pp. 849-858, Paris, France, September 16-20, 2000.

16. Rai, M. M., “Robust Optimal Aerodynamic Design Using Evolutionary Methods and Neural Networks,” AIAA Paper No. 2004-0778, AIAA 42nd Aerospace Sciences Meeting, Reno, Nevada, Jan. 5-8, 2004.

17. Rai, M. M., “Robust Optimal Design With Differential Evolution”, AIAA Paper No. 2004-4588, Tenth AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York, August 30th – September 1st, 2004.

18. Azarm S., and Li, W. C., “Multi-Level Design Optimization Using Global Monotonicity Analysis,” ASME Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 111, pp. 259-263, June 1989.

19. Corne, D., Dorigo, M., and Glover, D., Editors, *New Ideas in Optimization*, McGraw Hill , 1999.

20. Zitzler, E., Deb, K., and Thiele, L., “Comparison of Multi-Objective Evolutionary Algorithms: Empirical Results,” *Evolutionary Computation Journal*, 8(2), pp. 125-148.

21. Viennet, R., “Multi-criteria Optimization Using a Genetic Algorithm for Determining the Pareto Set,” *International Journal of Systems Science*, 27(2), pp. 255-260.

22. Perrone, M. P., “General Averaging Results for Convex Optimization, Proceedings of the 1993 Connectionist Models Summer School, M. C. Mozer et al. (Eds.), pp. 364-371.

23. Perrone, M. P., and Cooper, L. N., “When Networks Disagree: Ensemble Methods for Hybrid Neural Networks,” Artificial Neural Networks for Speech and Vision, R. J. Mammone (Ed.), 1993, pp. 126-142.

24. Rai, M. M., “Three-Dimensional Aerodynamic Design Using Artificial Neural Networks,” AIAA Paper No. 2002-0987, AIAA 40th Aerospace Sciences Meeting, Reno, Nevada, Jan. 14-17, 2002.

25. Rai. M. M., “Applications of Neural Networks in Design Optimization”, in preparation.

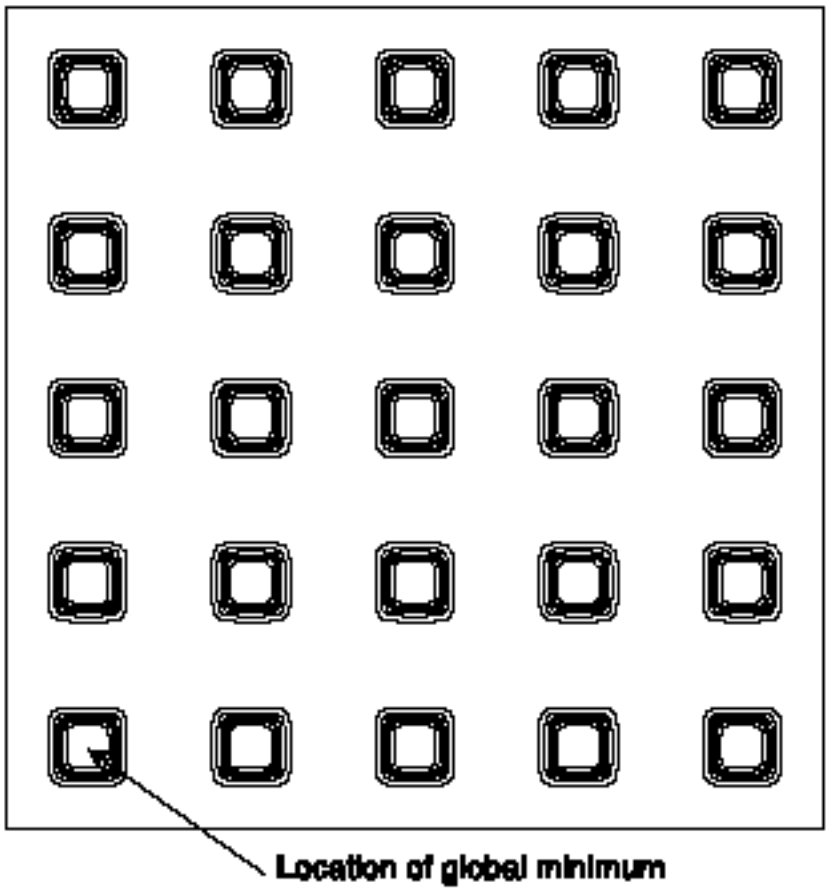

Fig. 1. Contours of De Jong's fifth function (Shekel's foxholes).

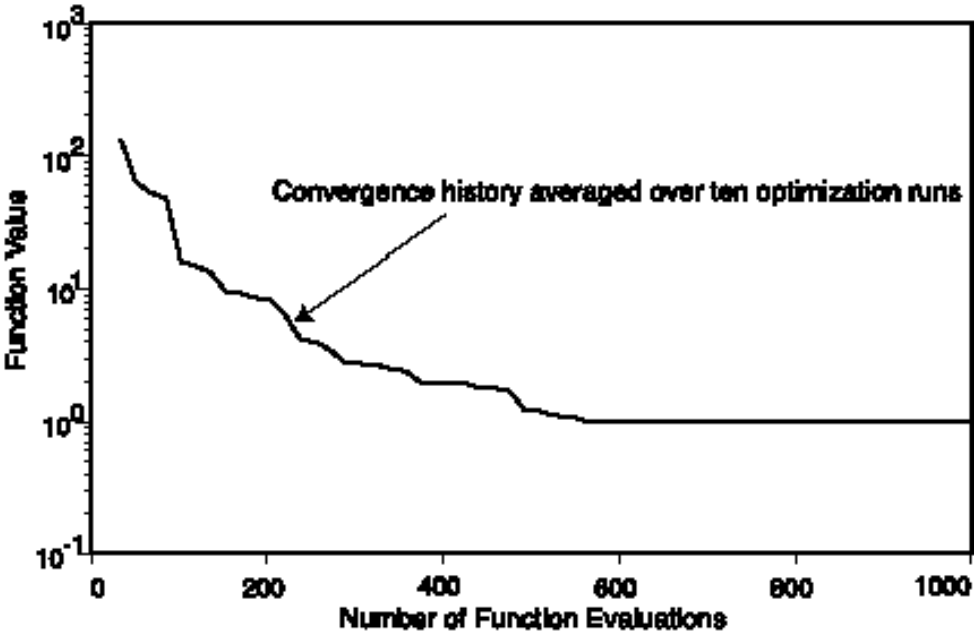

Fig. 2. Convergence history for De Jong's fifth function.

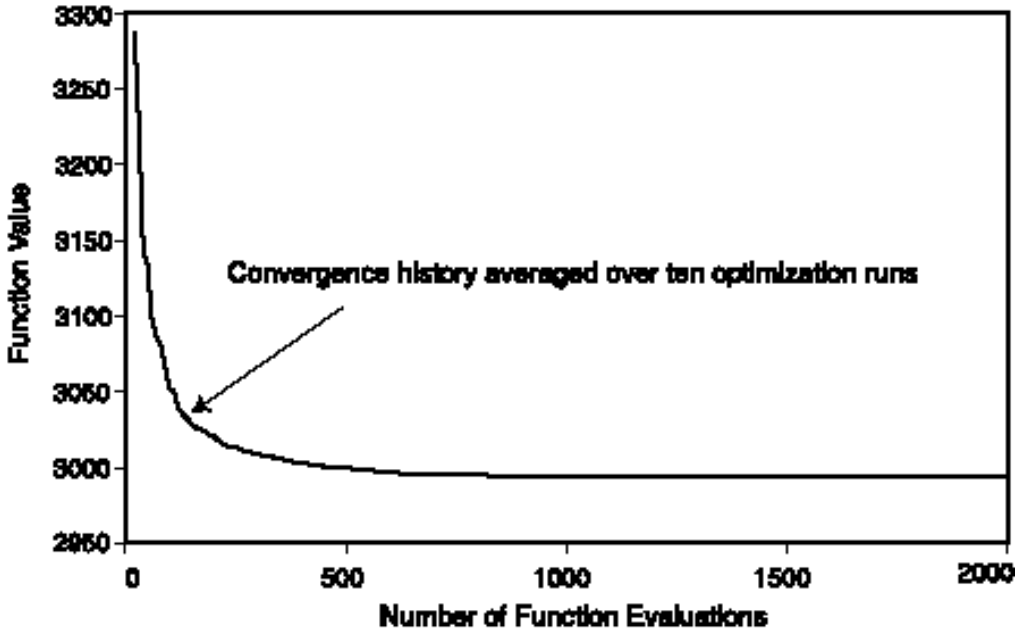

Fig.3. Convergence history for Golinsky's speed reducer problem.

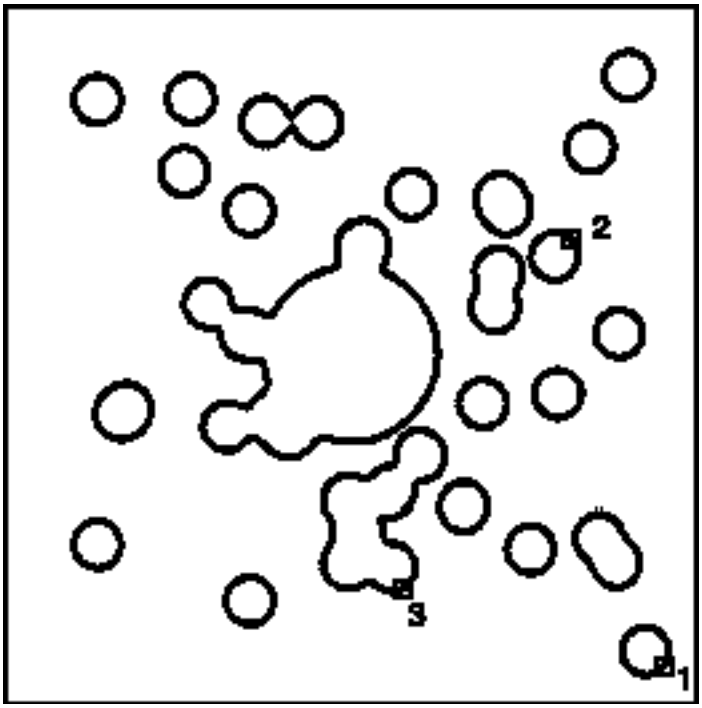

Fig. 4. Disjoint regions of feasibility with prioritized maxima.

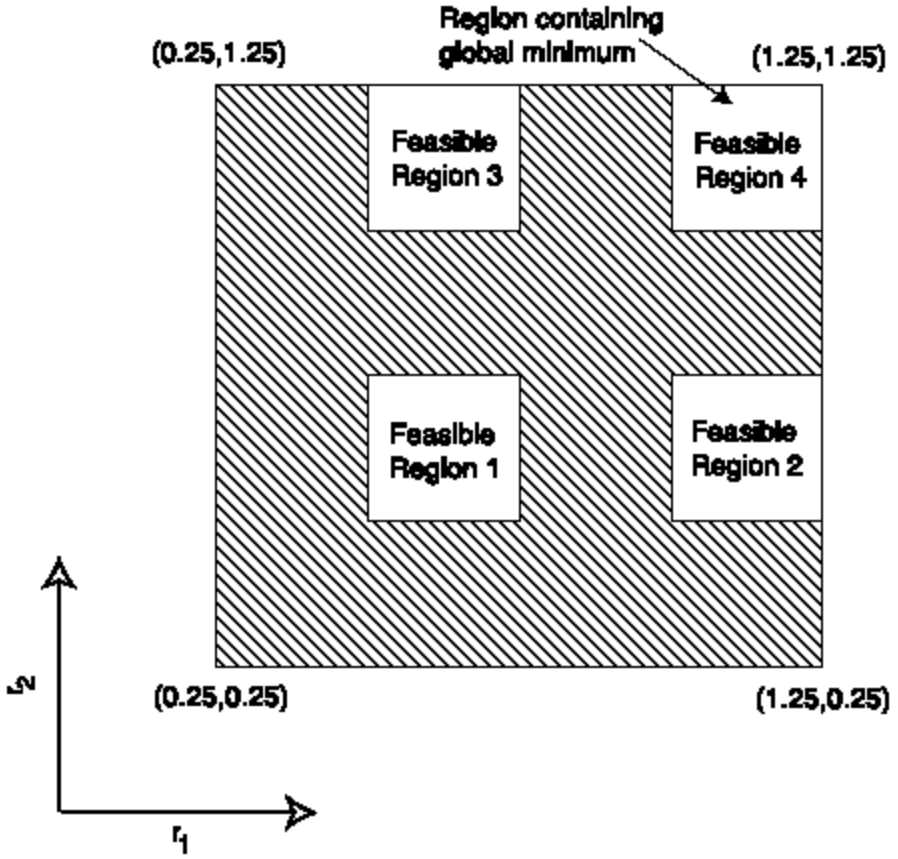

Fig. 5. Search region in two-dimensions showing feasible regions.

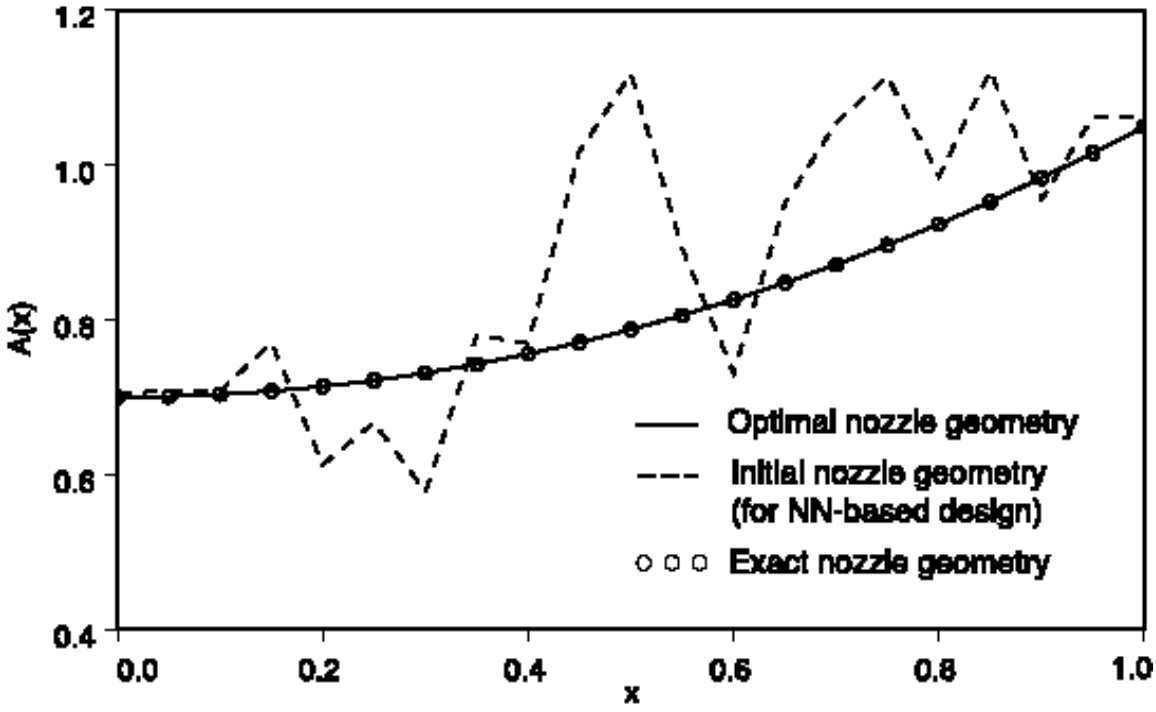

Fig. 6. Variation of the nozzle cross-sectional area in the axial direction.

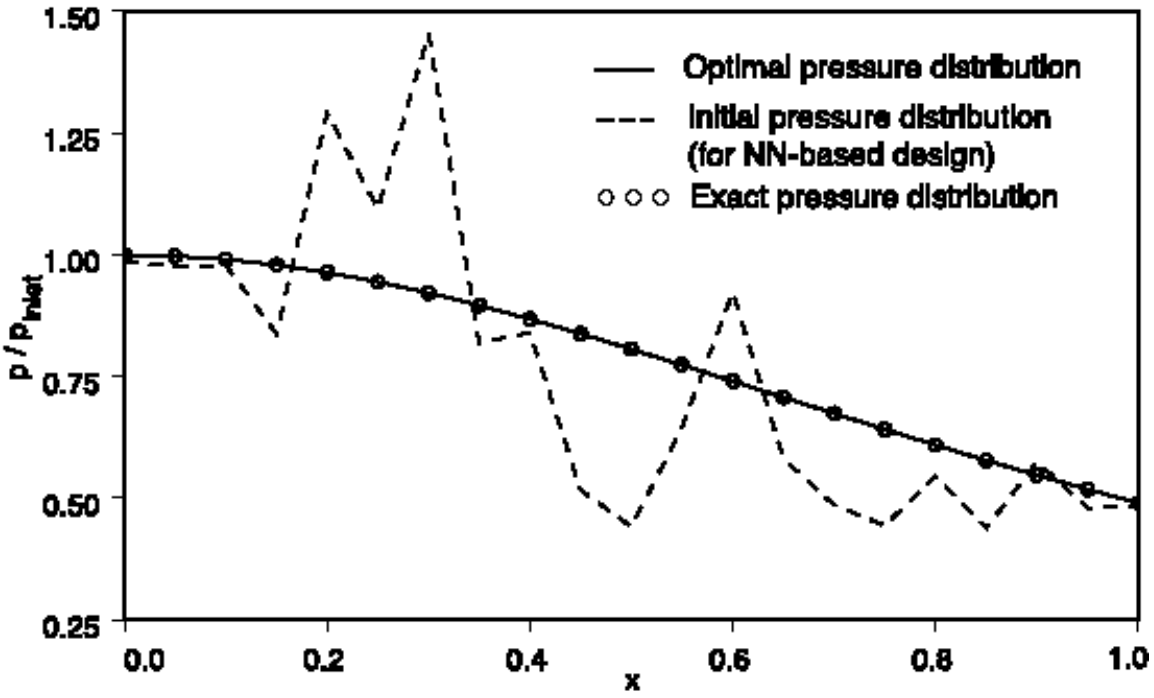

Fig. 7. Nozzle pressure distribution in the axial direction.

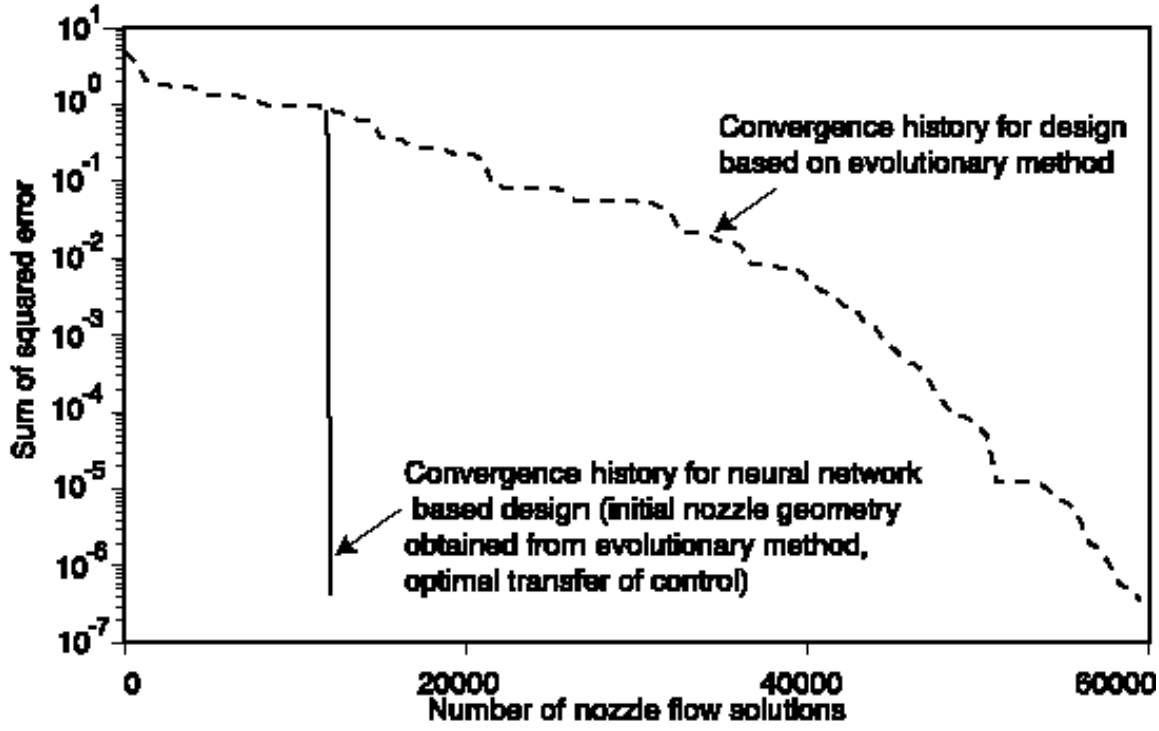

Fig. 8. Convergence history for the nozzle design optimization study (evolutionary method and hybrid method with optimal transfer of control).

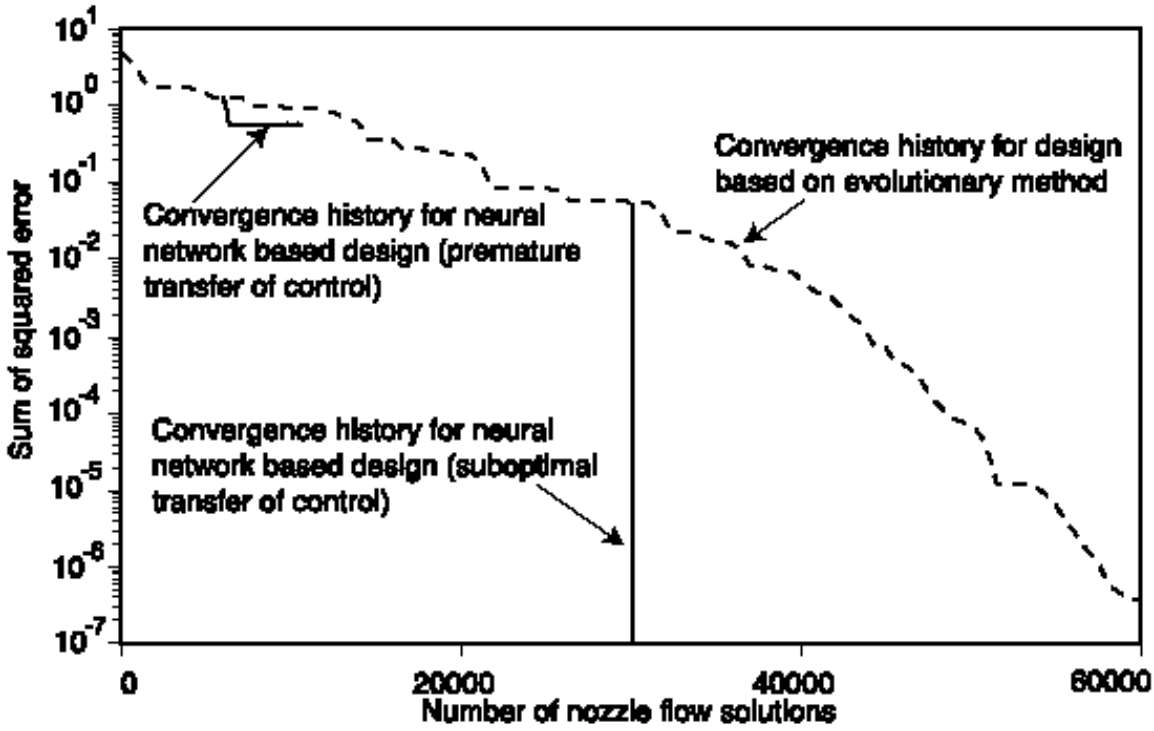

Fig. 9. Convergence history for the nozzle design optimization study (evolutionary method and hybrid method with premature and sub-optimal transfer of control).

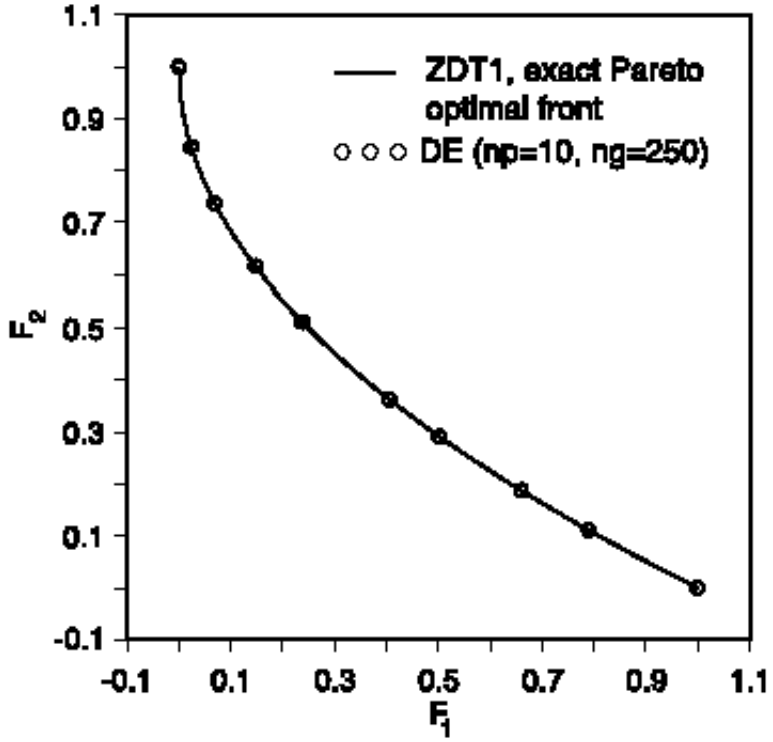

Fig. 10. Pareto optimal front in objective space for ZDT1.

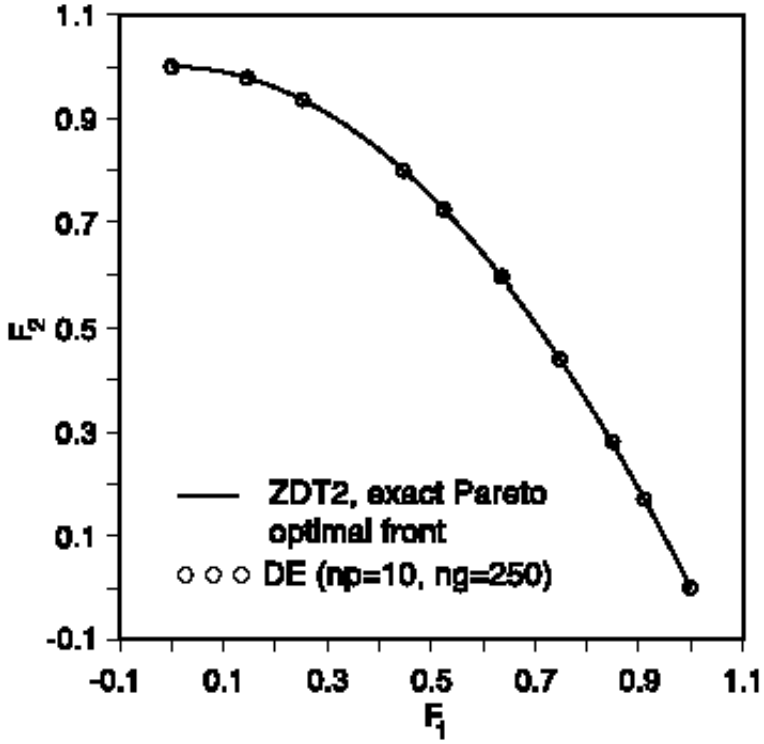

Fig. 11. Pareto optimal front in objective space for ZDT2.

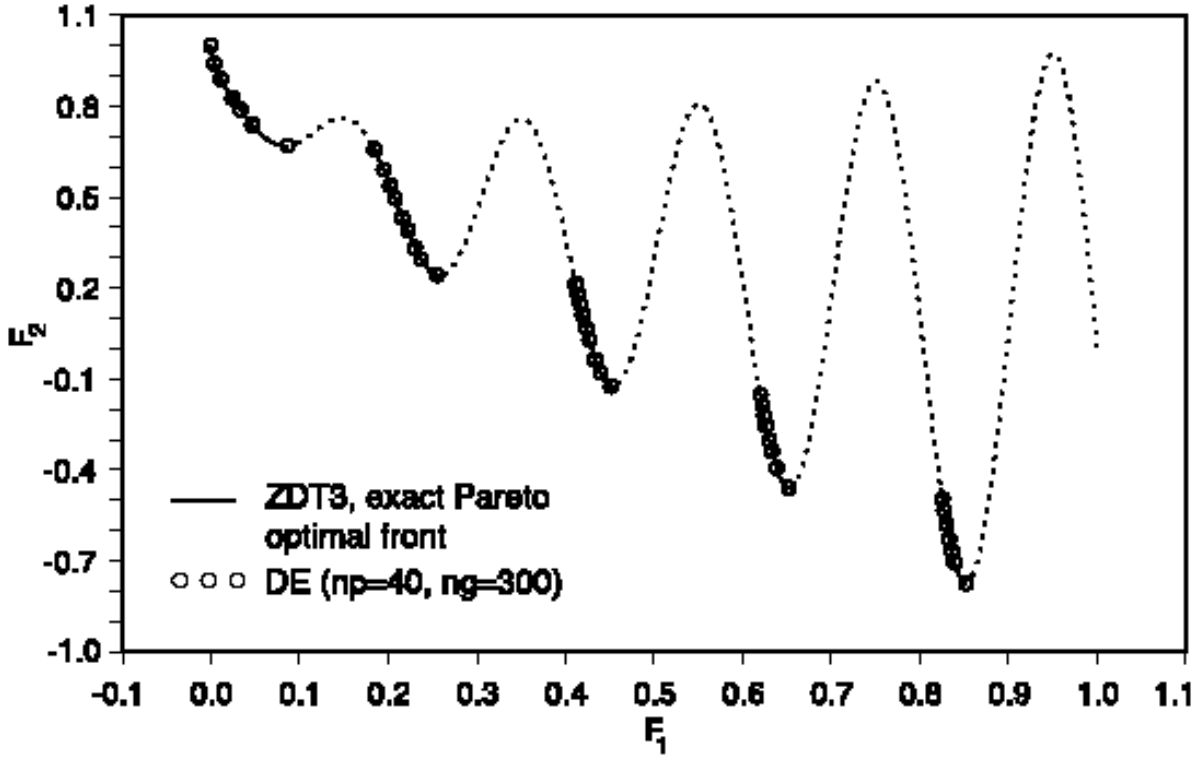

Fig. 12. Pareto optimal front in objective space for ZDT3.

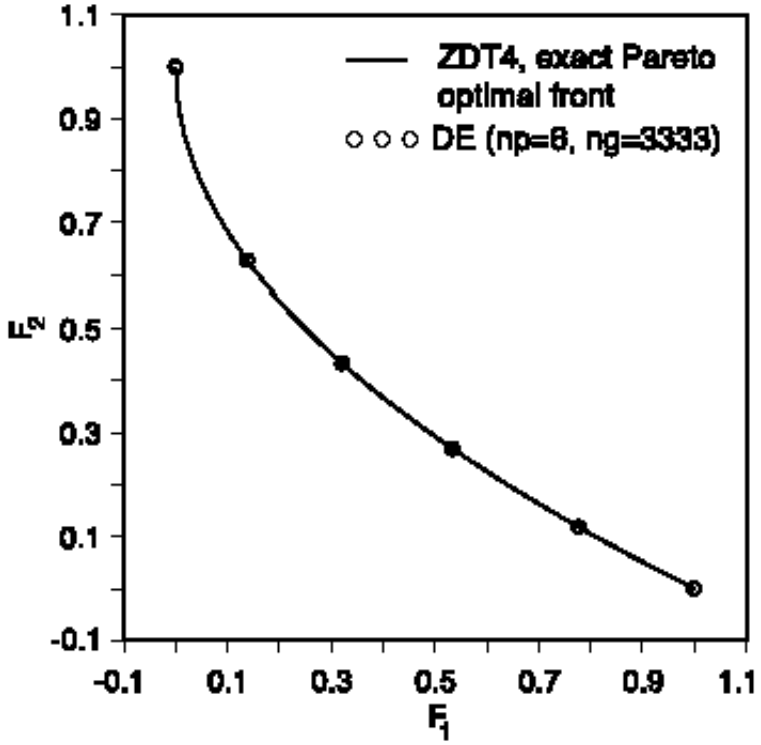

Fig. 13. Pareto optimal front in objective space for ZDT4.

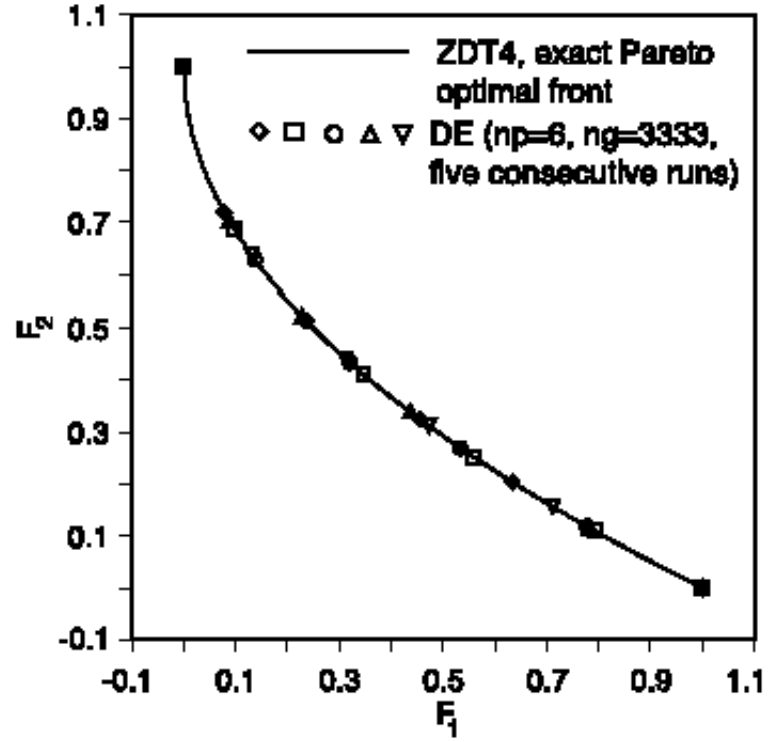

Fig. 14. Pareto optimal front in objective space for ZDT4 (multiple runs).

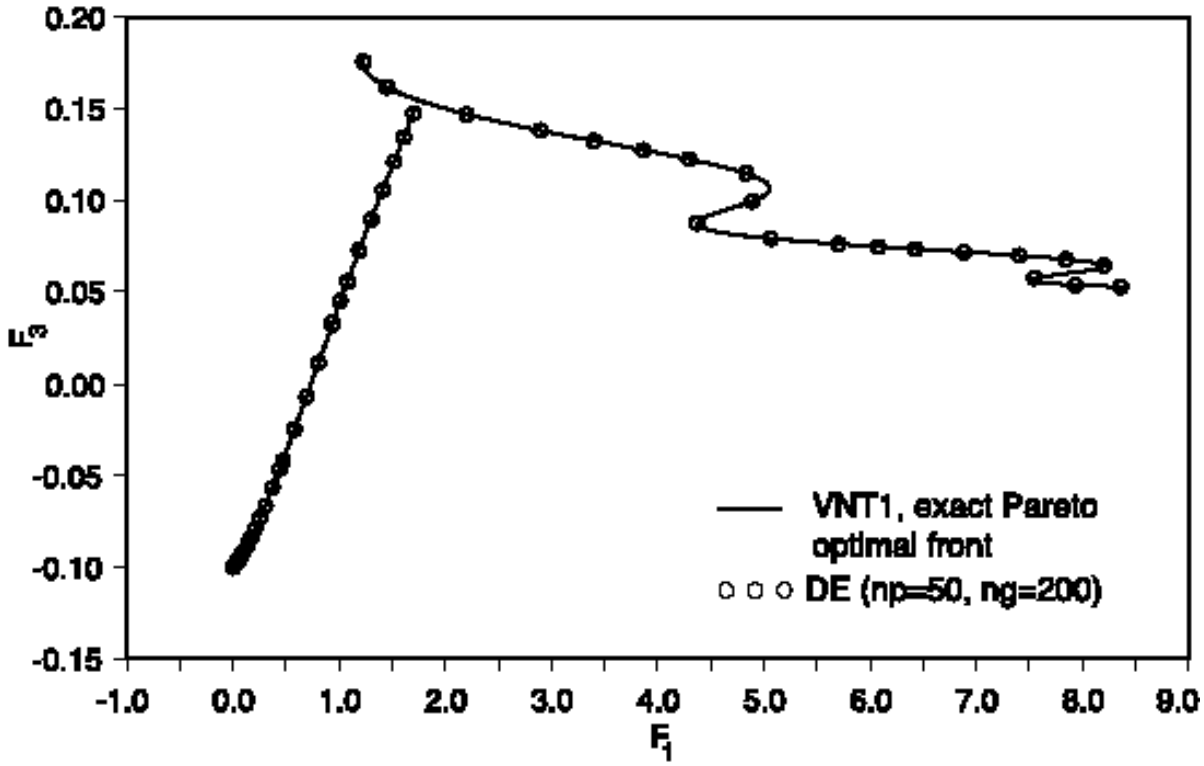

Fig. 15. Pareto optimal front in objective space for VNT1.

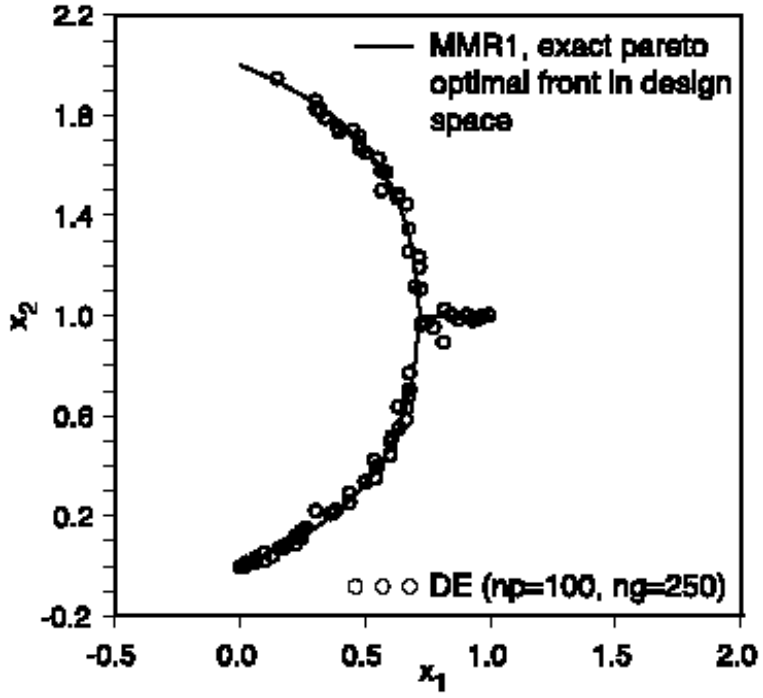

Fig. 16. Global Pareto optimal front in parameter space for MMR1.

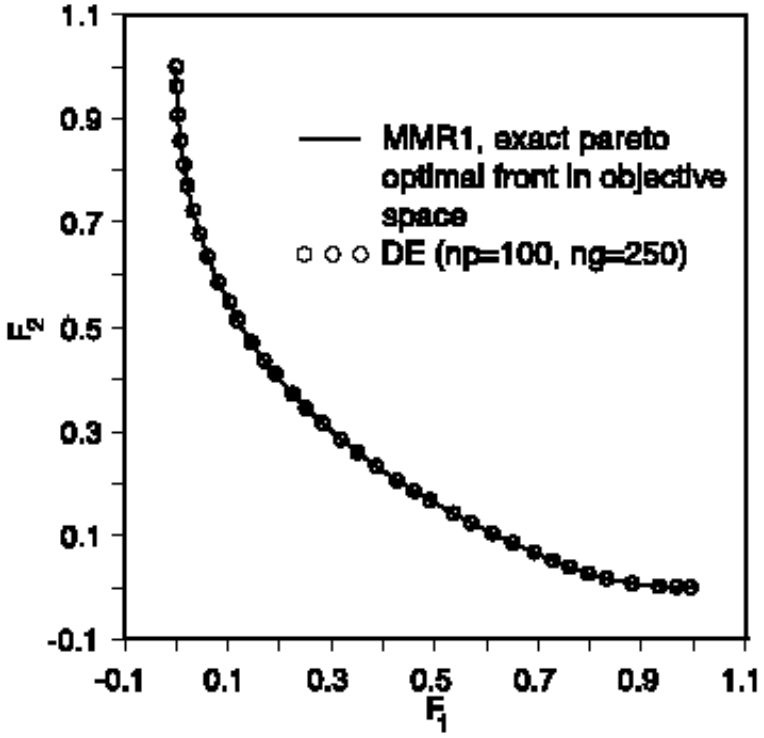

Fig. 17. Global Pareto optimal front in objective space for MMR1.

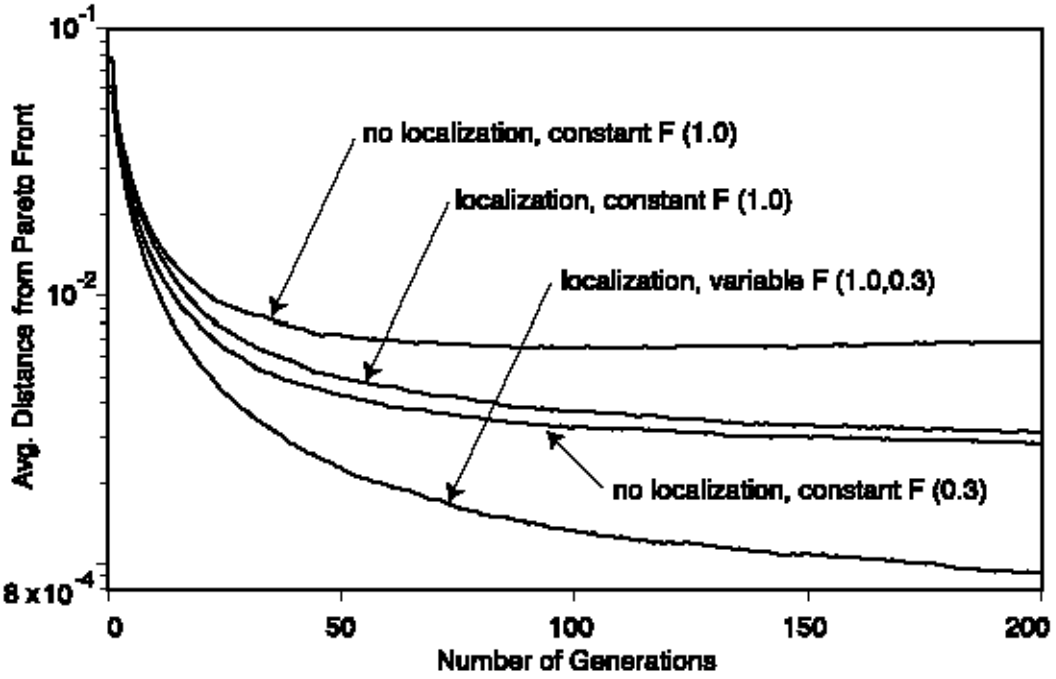

Fig. 18. Average distance from the global Pareto-optimal front as a function of the number of generations.

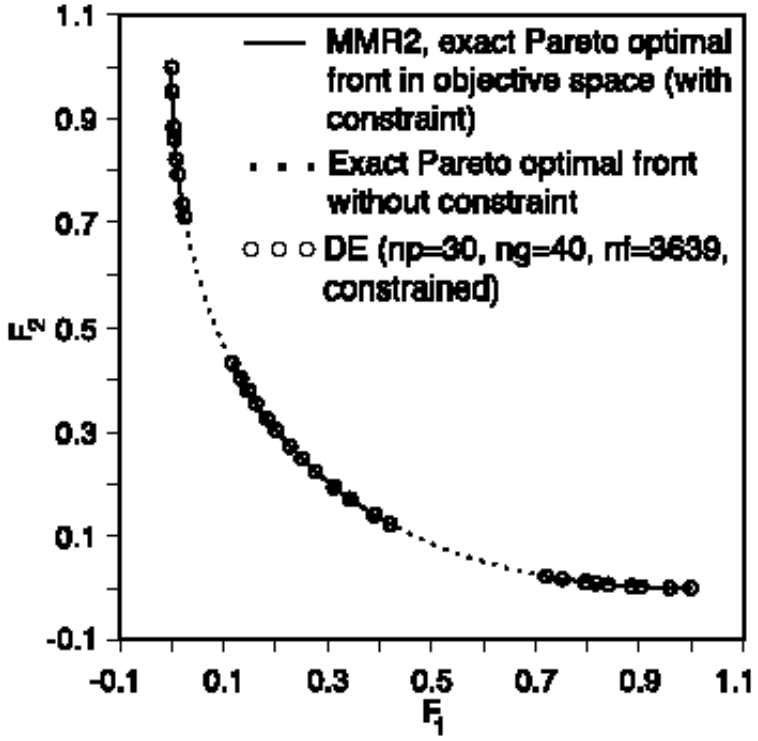

Fig. 19. Pareto optimal front in objective space for MMR2.

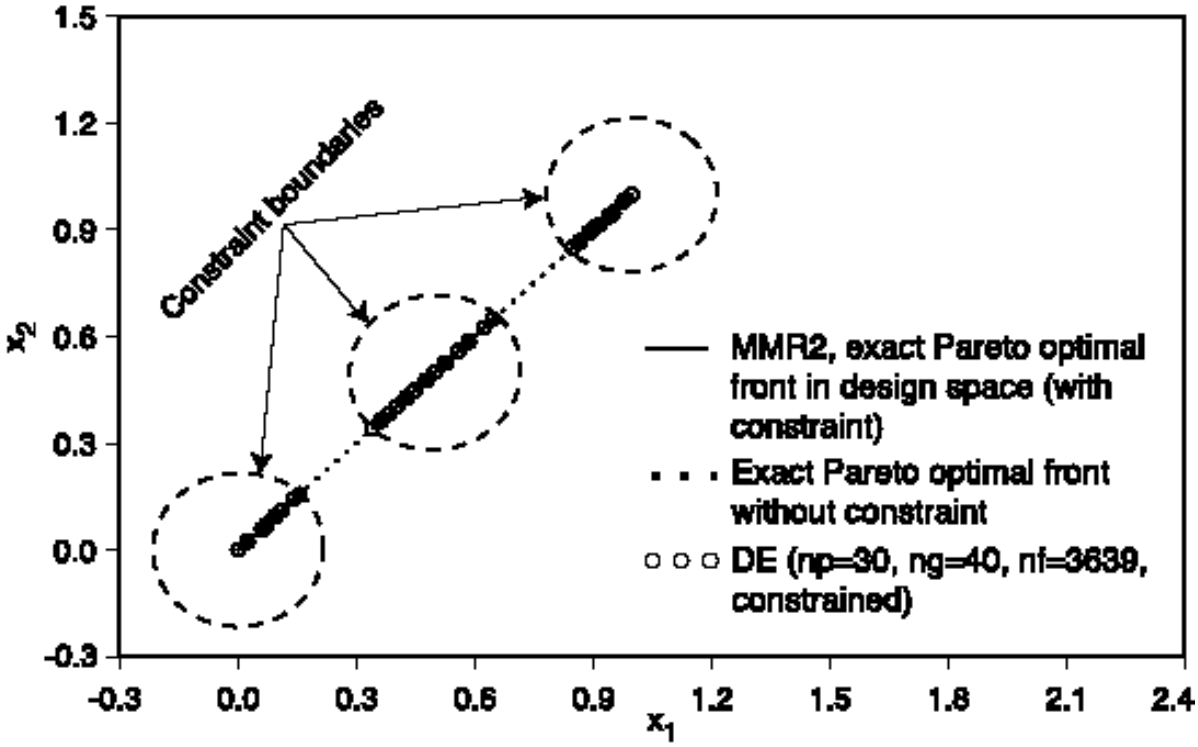

Fig. 20. Pareto optimal front in parameter space for MMR2.

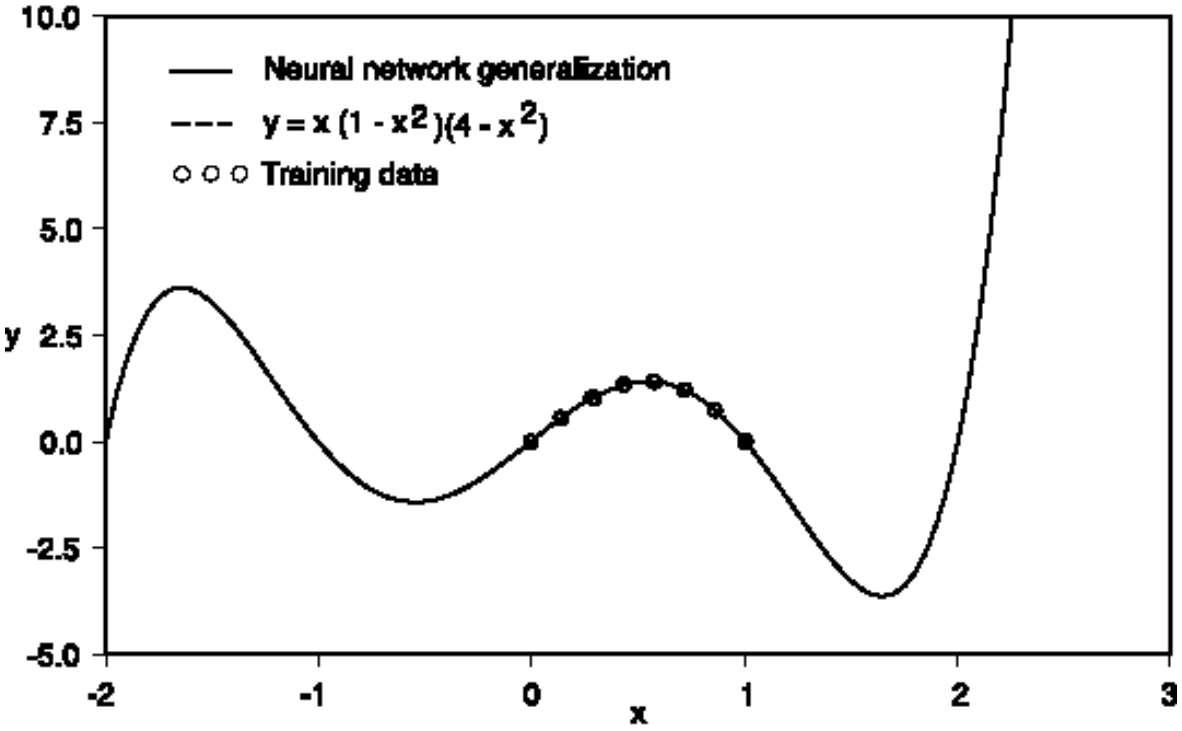

Fig. 21. Neural network generalization obtained for a fifth-order polynomial.

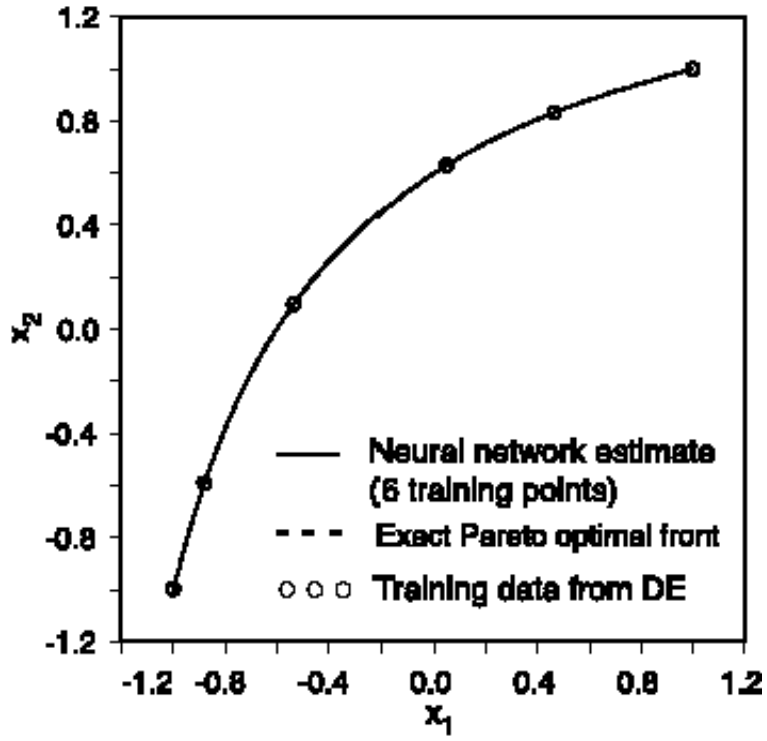

Fig. 22. Pareto optimal front in parameter space for MMR3 (fully converged data from DE).

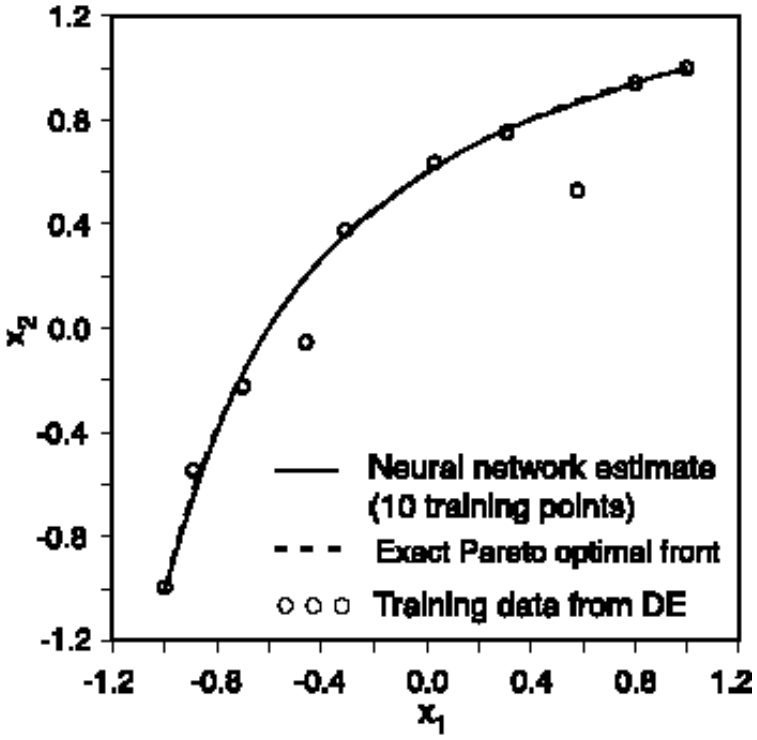

Fig. 23. Pareto optimal front in design space for MMR3 (partially converged data from DE).

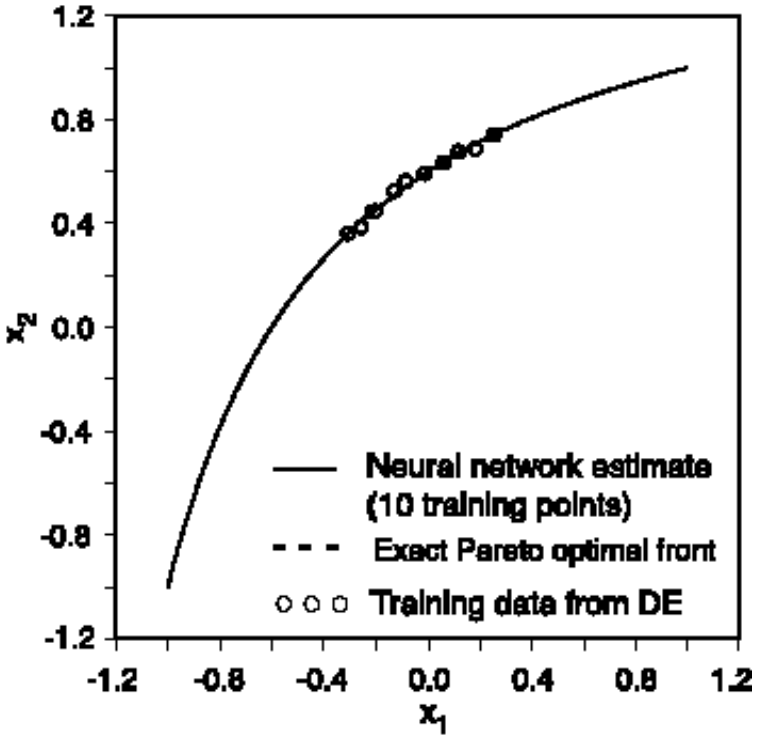

Fig. 24. Pareto optimal front in design space for MMR3 (restricted search domain).

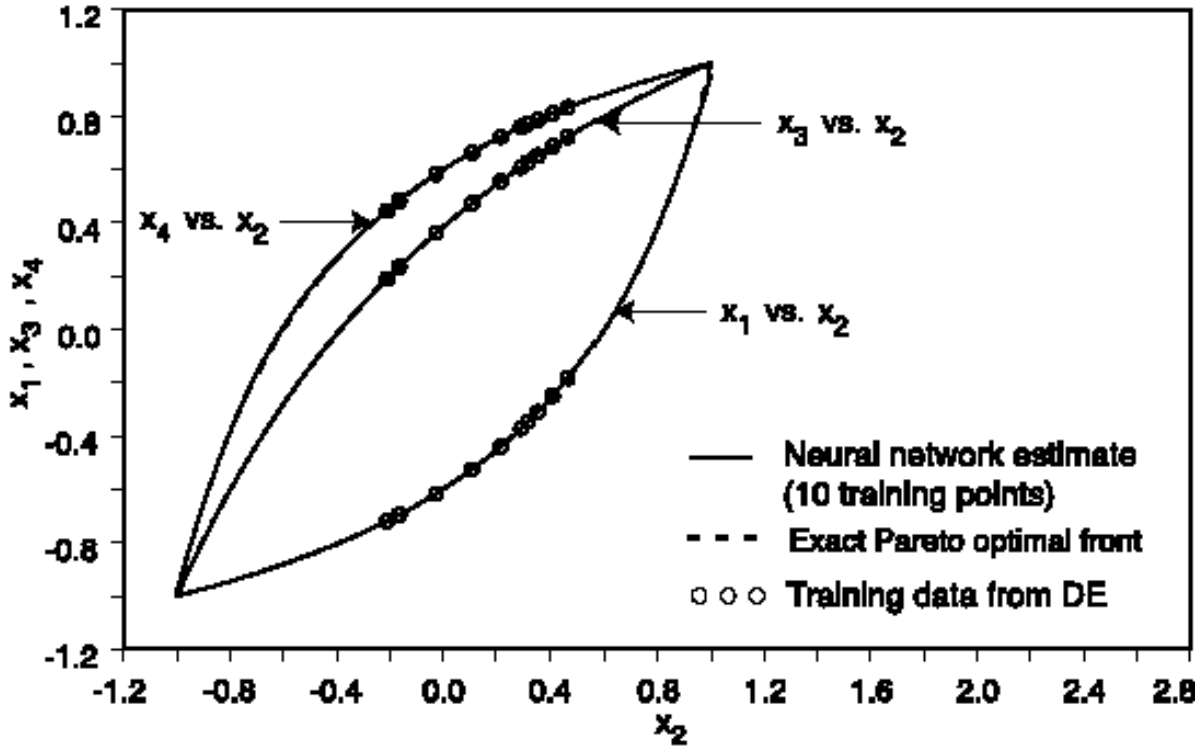

Fig. 25. Pareto optimal front in design space for MMR3 (four-dimensional search space).